Pedro Miguel Figueiredo Amaral
Mestre

Multipath inter-domain policy routing

Dissertação submetida para a obtenção do grau de Doutor
em Engenharia Electrotécnica e de Computadores

Orientador: Doutor Paulo da Costa Luís da Fonseca Pinto, Professor Catedrático, Faculdade de Ciências e Tecnologia

Júri:

Presidente: Prof. Doutor Virgílio António da Cruz Machado

Arguente(s): Prof. Doutor Manuel Alberto Pereira Ricardo
Prof. Doutora Susana Isabel Barreto de Miranda Sargento

Vogais: Prof. Doutor Paulo da Costa Luís da Fonseca Pinto
Prof. Doutor Luís Filipe Lourenço Bernardo
Prof. Doutor Rodolfo Alexandre Duarte Oliveira

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Junho 2012

Multipath inter-domain policy routing

Acknowledgements

Several people and Institutions have contributed to the development and writing of this dissertation.

First I would like to thank my advisor Prof. Paulo Pinto that contributed greatly to this work. Its personal effort to guide me and collaborate in my work was priceless and decisive for its success.

I would also like to thank Prof. Luis Bernardo for its valuable contributions and ideas. His views, often accompanied by a more detached look on the work where fundamental to the development of the initial ideas.

Msc. Students Claudio Assunção, Francisco Ganhão and Edgar Silva, all contributed with some work from their thesis to test some of the proposals. They were all a joy to work with.

I would also like to thank my colleagues from the Department of Electrical Engineering, Prof. Paulo Montezuma, Prof. Rodolfo Oliveira and Prof. Rui Dinis. All have at one point or another contributed with great professionalism to a good working environment. This has helped me to continue developing this work while performing my teaching assistant duties.

I am also thankful to all the financial support that helped me through this journey (travel funds, conference payments, etc.): CTS multi-annual funding project PEst OE/EEI/UI0066/2011; MPSat project PTDC/EEA-TEL/099074/2008; OPPORTUNISTIC-CR project PTDC/EEA-TEL/115981/2009; Fentocells project PTDC/EEA TEL/120666/2010.

Finally I would like to thank the Portuguese Science and Technology Foundation (FCT/MCTES) that has supported this work during a three years period through grant SFRH/BD/44476/2008.

Sumário

O problema do encaminhamento pode ser abstraído como um problema de descoberta de caminhos num grafo. Uma aproximação algébrica que descreva a forma como as rotas são calculadas e classificadas pode ser usada para modelar o problema. O cenário inter-domínio introduz novos desafios: o encaminhamento é feito entre redes geridas de forma independente; a classificação dos caminhos não é baseada em métricas mensuráveis mas sim em políticas; o encaminhamento deve ser baseado no destino e feito arco a arco.

Este trabalho parte do protocolo Border Gateway Protocol (BGP), identifica os seus principais problemas e a partir daí elabora sobre as características ideais para um protocolo de encaminhamento adequado à realidade inter-domínio. As principais áreas e contribuições deste trabalho são as seguintes:

- O estado de arte em modelação algébrica de protocolos de encaminhamento é usada para fornecer uma lista de condições possíveis que garantem o correcto funcionamento destes protocolos. Para cada caso são apresentadas as consequências em termos de optimalidade da solução e das restrições a impor à rede.

- Uma arquitectura de encaminhamento inter-domínio é apresentada. É feita a prova de que uma solução de encaminhamento multi-caminho é obtida em tempo finito e sem provocar "forwarding loops". São discutidas as vantagens e fraquezas da arquitectura.

- Um algoritmo de engenharia de tráfego é desenhado de modo a tirar partido da arquitectura proposta. Este funciona utilizando apenas informação disponível localmente e cooperação com ASes remotos para minimizar congestões na rede usando o mínimo de sinalização.

- Finalmente utiliza-se um modelo geral de um protocolo de encaminhamento baseado em políticas hierárquicas para estudar a eficiência do ponto de vista da operação do protocolo quando as condições de convergência são cumpridas. Esta análise resulta em algumas conclusões sobre como as políticas devem ser escolhidas e aplicadas de modo a atingir determinados objectivos.

Keywords: Ecaminhamento multi-caminho e inter-domínio, Modelos Algébricos, Teoria de Grafos, Engenharia de Tráfego.

Abstract

Routing can be abstracted to be a path finding problem in a graph that models the network. The problem can be modelled using an algebraic approach that describes the way routes are calculated and ranked. The shortest path problem is the most common form and consists in finding the path with the smallest cost.

The inter-domain scenario introduces some new challenges to the routing problem: the routing is performed between independently configured and managed networks; the ranking of the paths is not based on measurable metrics but on policies; and the forwarding is destination based hop-by-hop.

In this thesis we departed from the Border gateway Protocol (BGP) identifying its main problems and elaborating on some ideal characteristics for a routing protocol suited for the inter-domain reality. The main areas and contributions of this work are the following:

- The current state of the art in algebraic modeling of routing problems is used to provide a list of possible alternative conditions for the correct operation of such protocols. For each condition the consequences in terms of optimality and network restrictions are presented.

- A routing architecture for the inter-domain scenario is presented. It is proven that it achieves a multipath routing solution in finite time without causing forwarding loops. We discuss its advantages and weaknesses.

- A traffic-engineering scheme is designed to take advantage of the proposed architecture. It works using only local information and cooperation of remote ASes to minimize congestion in the network with minimal signalling.

- Finally a general model of a routing protocol based on hierarchical policies is used to study how efficient is the protocol operation when the correctness conditions are met. This results in some conclusions on how the policies should be chosen and applied in order to achieve specific goals.

Keywords: Inter-domain routing, Multipath routing, Algebraic models, graph theory, Traffic Engineering, Routing Problem

Contents

List of Figures

List of Tables

Acronyms

c2p	customer to provider relationship
p2c	provider to customer relationship
p2p	peer to peer relationship
p2patt	Peer-to-peer allowing transit traffic
p2pbk	Peer-to-peer allowing backup
AS	Autonomous System
ASes	Autonomous Systems
ASPATH	BGP attribute containing he AS path of a route
BGP	Border Gateway Protocol
CAIDA	The Cooperative Association for Internet Data Analysis
CDF	Cumulative Distribution Function
CIDR	Classless Inter-Domain Routing
DFZ	Default Free Zone
DHT	Distributed Hash Table
DNS	Domain Name Server
DTIA	Dynamic Topological information Architecture
EIDs	Endpoint Identifiers
FCP	Failure Carrying Packet
GDA	globally deliverable addresses
GRA	globally routable addresses
HLP	Hybrid Link State Path Vector Protocol

iBGP	external Border Gateway Protocol
iBGP	internal Border Gateway Protocol
ID	Identifier
IGP	Interior Gateway Protocol
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
ISPs	Internet Service Providers
LISP	Locator ID Separation Protocol
LOCAL_PREF	Local Preference BGP attribute
MED	Multi Exit Descriminator BGP attribute
MPLS	Multi-Protocol Label Switching
MRAI	Minimum Routing Advertising Interval
nam	Network animator for the network simulator version 2
NAT	Network Address Translation
ns2	Network simulator version 2
OSPF	Open Shortest Path First
QoS	Quality of Service
RIPE	European Regional Internet Registry
RLOCs	Routing Locators
RPSL	Routing Policy Specification Language
SLAs	Service Level Agreements
SPP	Stable Paths Problem
TCP	Transmission Control Protocol
TE	Traffic Engineering
UDP	User Datagram Protocol

Chapter 1

Introduction

Routing can be abstracted to be a path finding problem in a graph. Solutions to that problem have been the goal of much research from different areas of knowledge. The most common form of the problem is the *shortest path* problem, that consists on finding the path with the smallest cost where the cost is usually a numeric measurable metric. Some common solutions that solve the problem for *shortest paths* are well known and are the source of many of the most used routing protocols. The current reality in networking, and in particular in the Internet introduces some new requirements to routing.

The Internet is a large scale collection of independently owned and managed networks called domains or Autonomous Systems (AS, in plural ASes) [1]. Routing, when performed between such networks is generally defined as inter-domain routing. In this scenario each AS might have its own view of what path should be used, and the paths are selected based on a wider set of characteristics than a simple metric like distance or bandwidth. These characteristics are expressed using semantically rich concepts, usually called policies, that define the nature of the paths and their relative preference.

The *best path* problem is a generalization of the *shortest path* problem that is better suited to the policy based inter-domain scenario. The concept of the best path is richer because the ranking of the paths is not merely dependent on some numeric metric but instead on a set of desired characteristics such as the business relationships with other nodes along the path or some hierarchical ordering between them. Paths are therefore

[1] In this thesis we will use the terms "domain" and "Autonomous System" with the same meaning

calculated and chosen based on policy in what is commonly known as policy routing. Actually, the *shortest path* problem can be seen as a particular case of the *best path* problem. In that specific case the policy is the numeric metric that is assigned to links resulting in a weight for the path which is the sum of the links metrics, relative preference between paths is attributed according to the minimization of the weight.

Border gateway Protocol (BGP) is the *de facto* standard protocol used today in the Internet for inter-domain. It is a path vector protocol where nodes exchange routes [2] between them. These routes transport information about them in attributes that are part of the policy mechanism. BGP has a route decision process based on the routes attributes. This allows for the selection of paths to be controlled by setting the appropriate route attributes according to the desired routing policy. Typically, inter-domain routing is based on business related polices. For example, a route via a paid link to a neighbour (provider) is less preferred than a route via a link in which is the neighbour that pays (customer).

The current BGP based inter-domain routing architecture faces a number of challenges. The Internet has experienced a massive growth since it was designed not only in size but also in the number and variety of applications currently offered. This is causing scalability problems and has exposed some lack of capabilities of the system. BGP is extremely flexible for implementing different policies and ASes have exploited this flexibility widely with increasingly complex policies that many times can be conflicting. Each AS applies its policy on its own by manipulating route attributes and filtering which routes to advertise further. This highly distributed and flexible routing system can become quite complex and assuring its correct operation is getting increasingly harder. The fact that BGP was designed to only consider a single path amplifies all the problems generated by policy conflicts. Changes on a path have consequences on other paths in a cascading way. ASes are also becoming increasingly connected and the need to assure a high level of availability and cost effectiveness is causing an increased need to perform operations like defining backup paths or performing load balancing between different nodes. In a single path system such traffic manipulations have to be achieved by tweaking the routing state. This can have unpredicted outcomes in a network where nodes are uncoordinated

[2] Thoughout this thesis "route" and "path" are used with the same meaning

and forwarding is ideally done hop by hop based on the destination address. The overall flexibility of BGP that allows any kind of attribute manipulations further increases the problem. The complexity of the system also impairs scalability. For instance, the increasingly complex policies impair effective address aggregation and cause scalability pressure in the routing table sizes. The poor handling of the network dynamism by BGP is another major problem. Failures and other routing events cause lengthy re-convergence processes where the path exploration nature of the protocol is a drawback. Re-convergence is not even assured at all due to the BGP's complex route manipulations that can cause unpredicted behaviours in the presence of failures.

1.1 Problem

Inter-domain routing introduces several challenges: routing is based on policies that are semantically richer concepts than traditional metrics applied in *shortest path routing*; the network is built from independent nodes that are configured and managed independently so cooperation between them should be minimum and traditional distributed destination hop by hop forwarding is more suited than alternative methods like tunnelling or source routing; performing routing with rich uncoordinated and possibly conflicting policies is challenging because maintaining correctness is difficult and using single path routing protocols increases complexity with any desired traffic control having to be done by tweaking the routing state; finally, inter-domain networks are large scale and do not deal well with failures having a high number of messages to process and a slow re-convergence process.

This thesis proposes to solve the following problems. Define an inter-domain routing architecture that: supports policy routing based on business relationships suited for use in the Internet; provides multipath routing so that traffic control can be done by distributing traffic through the available paths and not only by changing the routing state; operates correctly using simple destination based hop by hop forwarding assuming that the configurations of the nodes are independent; takes advantage of its multipath nature to reduce the number of messages exchanged after failures and speed up (or even avoid) re-convergence.

An important part of the work consisted on proving the correct operation of the

proposed inter-domain architecture. Routing protocols can be modelled by using linear algebra and graph theory concepts. In the concrete case of this thesis the network is modelled by a directed graph and the applied policies by an algebraic structure that models the calculation and ranking of paths. The correct operation of the protocol for a given path calculation algorithm and forwarding method depends on the properties of the algebraic structure. The correctness convergence conditions are well known for *shortest path* routing protocols. However, in policy routing protocols some key properties are lost and the same conditions do not apply. Recent work in the area has modelled policy routing protocols and studied their convergence conditions. However, this work has been deeply connected to BGP. There as been also some work in the definition of the convergence conditions for the multipath case. Based on this work a list of the convergence conditions for a multipath policy based routing protocol using simple destination based hop by hop forwarding must be found to prove protocol correctness. The use of these models opens another interesting problem that is covered in this thesis: to find the practical implications of building such protocols. Examples of open questions are: what are the relationships between the needed correctness conditions and the implications on the operation of the protocol? Can such a protocol operate correctly with a high degree of available multiple paths, can all paths be valid? Can it operate correctly in any network or are there particular arrangements of sequences of nodes that cannot be present? In what conditions is a globally optimum solution possible? What is the impact in the amount of available paths for multipath routing and how does it impact the possible policies.

1.2 Contributions

This thesis makes the following contributions:

- Based on the current results in algebraic routing it provides a list of possible alternative conditions for the correct operation of multipath policy based routing protocols using destination based hop by hop forwarding. For each of the conditions the consequences in terms of the optimally of the solution and the need of restrictions on the network are presented.

- It presents the design of a routing architecture suitable for routing in the Internet. The architecture supports policy routing based on business relationships and features multipath routing using simple destination based hop by hop forwarding. The main characteristics are:
  - It is proven to work correctly in the Internet supporting a specified set of business policies.
  - Introduces multipath and provides a degree of separation between the finding of routes and the distribution of traffic.
  - Takes advantage of multipath to handle failures by reducing the amount of exchanged messages and avoiding long re-convergence times.
  - Scales to Internet-like network sizes. Advanced traffic control features like multipath routing or tweaking the traffic distribution do not increase scaling pressure.
- A Traffic Engineering (TE) protocol for congestion control is defined for operation on top of the new routing architecture. This protocol is suited for the inter-domain scenario, uses minimal coordination between nodes and only locally available information.
- A study on the practical implications when implementing a policy routing multipath protocol that operates correctly. A general model of a routing protocol based on hierarchical policies is used to analyse the impact of meeting the correctness conditions on the protocol's operation. This provides insights in how the policies should be chosen and applied according to specific goals for a protocol.

1.3 Organization

The thesis is organized as follows. Chapter 2 presents some background concepts about the inter domain scenario and the current BGP based architecture for the not so familiarized reader followed by a small introduction to some algebraic concepts used in the thesis. Chapter 3 reviews the open issues in inter-domain routing and presents the existing literature on the subject covering proposals to solve each of the identified issues. This work

leads to the identification of the best characteristics for an inter-domain routing protocol. In Chapter 4 the existing formal models used to model BGP are presented and their relationships with a pure algebraic modelling are shown. Then, it is discussed how a routing protocol can be modelled algebraically and how the existence of a solution for the algebraic routing problem relates to the correct operation of a protocol. We then reach a series of possible conditions for correct operation based on existing work in algebraic routing. We then show the relationship between several known results and reach a set of conditions for correct operation of multipath routing protocols using destination base hop by hop forwarding, Chapter 5 describes our proposal for an inter-domain routing architecture called DTIA: a formal model is presented and used to prove its correctness. In Chapter 6 a traffic engineering protocol that avoids link congestion with a very lightweight signalling designed to operate on top of DTIA is presented. Chapter 7 presents the results of some experimental simulations to study some of the DTIA characteristics. They cover scalability, failure handling and the traffic engineering protocol. Chapter 8 contains a generic discussion for multipath routing policy protocols about the trade-off's and implications of designing such protocols. It relates the convergence conditions and the chosen policies with practical issues like the number of usable multiple paths, the need to invalidate paths or the restrictions that need to be applied to the network. Finally, Chapter 9 concludes the thesis work by presenting a discussion on the thesis contributions and pointing to some possible future work.

Chapter 2

Background

This chapter describes some major concepts of BGP, and some basic algebraic concepts needed to understand the algebraic models used throughout the thesis. BGP is the *standard* inter-domain routing protocol and understanding its operation is fundamental in order to analyse the current state of the art and the problems to solve. In the last section of the chapter we introduce some linear algebra concepts to familiarize the reader with the terms used in the algebraic routing models.

2.1 The inter-domain scenario

The Internet inter-domain routing consists in exchanging routing information between separate and independently managed networks called Autonomous System (AS) with the objective of establishing paths amongst them. ASes contain several internal routers and deal with the distribution of internal routes. This is called intra-domain routing. The focus of this thesis is the inter-operation of the ASes to exchange routing information between them. ASes are commercial entities with given business goals. They can be internet service providers (ISPs); content providers; transit operators; or simple customer ASes like a small company or university network seeking connectivity to the Internet. These different roles influence the types of contracts that are celebrated between ASes when they decide to connect between them. These contracts (and their costs) are decisive for the way routes are exchanged between the ASes. The decisions of which routes to exchange and which routes to prefer are commonly know as *routing policies*.

2.2 Common business polices

In the Internet we can easily identify some common business relationships [Gao00], which form the basis of inter AS routing exchange policies. The three most common relationships are:

- provider to customer ($p2c$) relationship - In this relationship one of the ASes, the provider, delivers all the traffic received from the customers and receives a payment for that.

- customer to provider ($c2p$) relationship - This relationship is the dual of the previous one.

- peer to peer ($p2p$) relationship - In this relationship both ASes agree on exchanging traffic coming from their clients and going to their clients. Most of the times there are no payments involved but it depends very much on the relative dimension. For instance, a content provider (sends more traffic than it receives) might want to peer with an eyeball provider (a provider having many customers who download the content) in order to be more popular. The eyeball provider might charge for this peer link.

These business relationships influence how routes are distributed. They lead to two common routing practices: the first concerns the preference of routes according to financial cost, meaning that routes learnt from customers are preferred to routes learnt from peers and these are preferred to routes learnt from providers; the second is related to route announcements. For financial reasons an AS does not advertise routes learnt from providers or peers to other providers or peers. The reason is simple. This would allow transit traffic to flow through it without revenue. For example, traffic for a provider going to another provider would mean that an AS would be paying to transport third party traffic.

Figure 2.1 illustrates these relationships. AS B and AS C are peers meaning that they exchange routes to F and E. AS C does not export routes learnt from AS D to neighbour AS B, because this would mean that it would carry transit traffic between B and D. Since this makes no commercial sense these routes are not exported. We refer to such routes as

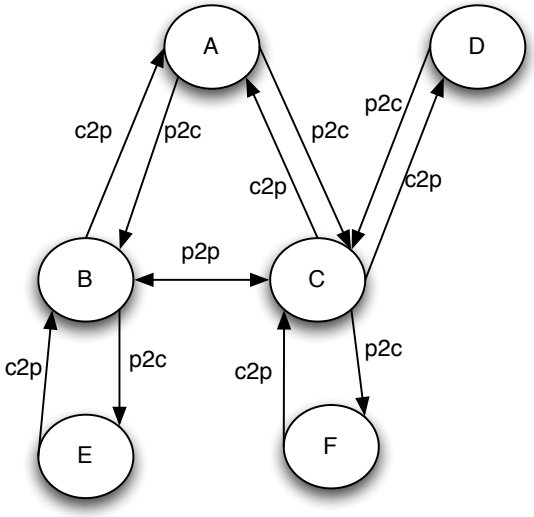

Figure 2.1: Example AS topology

invalid routes. Another invalid route in the example is the route A-C-D, since it would mean that AS C was carrying transit traffic between A and D when it pays for both links. There are more complex routing relationships between the ASes, like sibling relationships between ASes managed by the same entities or backup relationships that allow transit traffic outside the usual conditions, for failure affected scenarios.

2.3 Inter-domain routing with BGP

As stated above, BGP is a path vector protocol that is currently in its fourth version. It works by exporting reachable paths for visible destinations. Routers receive paths and after a filtering procedure select the ones that they intend to advertise. Then they prepend their identification and forward the path advertisements to the valid neighbours. Routes are exchanged between neighbours via TCP sessions. There are two types of sessions: i-BGP sessions, that are used to exchange information about routes between the border routers running BGP and the routers inside the AS; and e-BGP sessions that export routes between domains. The routes are organized in 3 tuples: destination address; the AS level path and the path attributes. The AS level path is an ordered list of the ASes traversed to reach the destination and is used to avoid routing loops. The attributes are used in the filtering and decision process that elects the *best path* for each destination. The *best path*

is then used for forwarding and it is advertised to neighbour ASes.

2.4 Decision process

BGP's decision process is based on a set of rules that prioritizes the importance of the route attributes. Table 2.1 summarizes the decision rules in order of priority and it indicates who sets the corresponding attribute values: either the router making the decision (Local Router) or the neighbour from which it received the route announcement (Neighbour).

Table 2.1: BGP decision Process

#	**Rule**	**Who defines the Value**
1	Highest Local Preference (LOCAL_PREF) attribute	Local Router
2	Lowest AS Path length	Neighbour
3	Lowest Multi Exit Discriminator (MED) attribute	Neighbour
4	eBGP over iBGP	Neither
5	Lowest Interior Gateway Protocol (IGP) cost	Local Router
6	Lowest Router ID	Neither

Rule 2 is a similar approach to the traditional shortest path routing, as the path with the least number of ASes is preferred. Rules 1 and 3 are more interesting because they are good examples of how complex routing policies can be performed using BGP. The LOCAL_PREF attribute is a good example of how an AS can control outbound traffic preferring a path that is not the one with the shortest AS path. The MED attribute is used to try to influence the decisions in neighbouring ASes. It is a measure of how a route should be discriminated in the decision process of the neighbouring AS. Note that the LOCAL_PREF attribute at the neighbour can of course overrule this rule. Besides the manipulation of some of the attributes by an announcing AS, routes can be suppressed (not announced at all) or aggregated.

2.5 Algebraic concepts

Linear algebra concepts have been used for several scientific purposes. Traditional algebraic structures like groups, fields, or rings are not the most appropriate tools for modelling and problem solving in specific areas. What is the more appropriate are some more prim-

itive structures. A very easy example is the case of arithmetic, the algebraic structure

$$(\mathbb{N}^{0+}, \times, +)$$

made from the set of positive integers endowed with the ordinary addition and multiplication. This structure does not have all the properties of a field or even of a ring and is still of interest. For the modelling of routing there are also more primitive structures that are of interest [GM08]. The main reason is that the operations of routing (extending paths and deciding on priorities) usually do not have so complete properties as the ones imposed in groups, fields or rings. The more primitive structures are formed by a set endowed with two binary operations $\otimes$ and $\oplus$. One example is the well known modelling of the *shortest path* problem in a graph. The solution for this problem can be written as linear equations in the algebraic structure

$$(\mathbb{R} \cup \{+\infty\}, \text{Min}, +)$$

which is the set of reals endowed with the operation Min (minimum of two numbers) corresponding to the $\oplus$ operation and the addition which is the $\otimes$ operation.

With this in mind we will consider generic algebraic structures of the form:

$$(E, \oplus, \otimes)$$

In the particular case where the operation $\oplus$ induces an order in the set E we can have a structure in the *ordered* form with:

$$(E, \preceq, \otimes)$$

where the order $\preceq$ is given by $\oplus$ as: $a \preceq b$ if $a \oplus b = a$ and this is typically the case in routing models where the $\oplus$ operation is the Max or Min operations. Other examples of such algebraic structures are:

- $(\mathbb{R}, Min, +)$ or $(\mathbb{R}, Max, Min)$ that model the *shortest path* problem and the *maximum capacity* path problem, respectively.
- $(0, 1, Max, Min)$ or *Boolean Algebra* the algebraic structure underlying *logic*.

Chapter 3

Inter-domain routing

3.1 Introduction

This chapter presents the state of the art in the area of inter-domain routing. The BGP based approach faces a number of challenges [YMBB05], mainly due to the explosive growth of the network both in size and in the amount and variety of applications available today in the Internet [LIJM+10]. The Internet is growing both in the number of ASes and in the interconnections between them [EKD10]. Also the network was not originally designed with many of the current services in mind and it is not prepared to handle their needs. As stated before one of the problems this thesis aims to solve is to provide an inter-domain routing architecture that focuses on five main issues: scalability, failure handling, multipath routing, traffic engineering and finally convergence and policy compliance. There are other important issues currently under discussion in the literature, such as security or inter AS routing distribution problems [YMBB05] but we decided to not include them in order to restrict to the scope of the problem we aim to solve.

In architectural terms, the reality can be characterized in the following way: BGP became widely established and we have witnessed that no fundamental parts of the overall inter-domain routing architecture have ever been changed. Nevertheless, we can divide the research being made into incremental, and new routing architectures approaches. In the second category there are radical clean slate approaches that are not bounded to the current reality, and approaches, like the system of this work described in chapter 4, that propose new routing architectures (without using BGP) but still maintain the current

Internet organization and business paradigm.

This chapter presents the current solutions employed by BGP for each of the five issues followed by a description of some other relevant work reported in the literature.

3.2 Scalability issues

There are two main scalability issues: the increasing size of the routing table size and the increasing rate of BGP updates (churn). The increasing of the routing table size is related mainly to architectural questions like topology evolution, the addressing scheme (that mixes the location and the identification of nodes and currently it starts to be seen as an handicap) and the way in which the addresses can be aggregated. The increasing of churn is related to the way failures are handled in BGP. Both issues are also inter related and it is remarkable how the increasing numbers of prefixes in the routing tables can increase the needed messages to respond to a routing event up to a point that jeopardizes the routers ability to deal with them. In this section we focus on the increasing size of the routing table leaving the churn rate aspect for the section concerning failure handling.

The global routing tables in the Internet have been growing in size over the years. The main reason and probably the most difficult one to avoid is the explosive growth in size that the network has suffered along the years. It is known that traditional routing protocols like BGP cannot have a better than linear growth of the routing table with the number of addresses to route [KcFB07].

A traditional solution to improve routing scalability is to use address aggregation. A first well-known case was the introduction of Classless Inter-Domain Routing (CIDR) in the mid nineties. It reduced the size of the routing tables by aggregating addresses into a single prefix. However, in the current network this aggregation is not always possible. There are two main factors contributing for this.

The first is related with the evolution of the network. Sites are increasing their connectivity with stub ASes being connected to more than one provider in a practice known as multihoming. Multihoming impairs address aggregation. Consider an AS getting its prefixes from provider 1 and having other n providers. Every provider but provider 1 cannot aggregate the prefixes since they do not belong to them. Even the one provider

that can may not want to aggregate – if it does it might get no traffic because more specific longest match paths are preferred. Finally, if an AS gets a set of provider independent prefixes then no aggregation is possible.

The second factor is the possibility of using several links from a multihomed AS to perform load balancing. Since BGP is prefix based the natural way to achieve the desired routing behaviour is by separating prefixes and performing advertisements such that traffic can be received from different providers to different prefixes [YMBB05]. According to [BGT04] prefix de-aggregation due to load balancing and multihoming were the two faster growing contributors for route table growth at the time of the study.

Taking in consideration the arguments just described the scalability of the routing table is an inherent problem of BGP, although there are some efforts trying to prove that the problem is not getting too serious over the years.

3.2.1 Literature

For IPv4 there are not many solutions to the routing table size problem. One typical solution to solve the poor aggregation is to filter prefixes that are excessively long but this can lead to some routes not being redistributed in the global Internet. Other existing solution is cooperation between the providers where the AS is multihomed. In this method the providers are interconnected and the secondary provider only announces the customer routes to the primary AS. All traffic from the Internet to the customer will flow to the primary provider. Aggregation is guaranteed since the secondary AS (that cannot aggregate the customer prefixes) does not advertise them. This solution has several drawbacks like the need of cooperation between two different providers and the absence of fault tolerance for the customer-primary AS link as well as for the link between the providers. Network address translation (NAT) can also be used. In this case prefixes from both providers are assigned to the customer and a NAT box in the edge translates the address (if necessary) to an address of the space of the currently working provider. NAT is not considered a good engineering solution for IPv6 networks but it can be a good transition solution for IPv4 [DLB06]. However, it has a single point of failure (the NAT box).

The aggregation issue is better dealt with when using IPv6, and in that case there are

several solutions. There are three approaches: routing, middle box and host centric. The host centric approach is the most promising one. It implies the use of several provider aggregatable addresses per host, ensuring aggregation but with an impact on how routing and traffic engineering is performed. The basic principle is that each host has an IPv6 address for each provider its AS is multihomed to. Aggregation is then guaranteed if the multihomed AS only advertises to each upstream provider the set of addresses that belong to it. It is also necessary to apply source based routing in the exit of the customer AS in order to deliver the traffic to the right upstream AS according to the address the host is currently using. Using multihoming with multiple prefixes leads to the increase of the number of paths available to the hosts [dLQB06] and can provide some base for Traffic Engineering in the selection of the path to use.

Some theoretical work has been done to find schemes that can grow sub-linearly with the number of routing addresses. This approach is called compact routing [KcFB07]. They are based in schemes where the need for a better path, whatever the metric, is relaxed. The routing performance is measured by the difference between the resulting paths weights and the best paths available. Good routing performance is achievable with such approaches with logarithm growth of the routing table in relation to the number of routing addresses [KcFB07], [Nor09]. However, the sub-linearity of the routing table size growth is lost for topology independent addressing and more importantly they do not work well in dynamic topologies[KcFB07]. These characteristics make these approaches hard to implement in Internet like networks, or any network where dynamism is important. A solution for a non linear growth of the routing tables is not in the focus of this work, and research in the compact routing area suggests that a long way has still to be done to apply it to non static networks with topology independent addressing.

Since a sub linear routing table growth is still very challenging a number of solutions try to enhance scalability providing better aggregation of addresses and/or some sort of separation in the address space.

In the current BGP based architecture a unique space is used to identify and locate a network node. Separating both functions and relying on a distributed mapping service should alleviate scalability problems. The Locator ID Separation Protocol (LISP) [D.07]

attempts to separate both functions, through an IP over UDP tunnelling approach. It separates the address space in two types: locators (usually border routers) called Routing Locators (RLOCs) and end systems (hosts) called Endpoint Identifiers (EIDs) . As expected on an Autonomous System, end hosts identified by EIDs could be served by several border routers, identified by RLOCs on their domain. The basic concept of LISP is to tunnel packets through the Internet's core, from the RLOC of the source EID to the RLOC of the target EID. Routing is therefore based on the target RLOC. If only RLOCs are exported to the Internet's core, the number of BGP entries is reduced. However, to route packets from the source EID to the destination EID, RLOCs are supposed to cache a mapping between RLOCs and EIDs. A set of messages and procedures (subsets, priorities, etc.) is defined to implement the mapping service. LISP [D.07] describes four variants for the role of the routers in the service and the highest one (LISP 3) assumes that an external service performs the mapping. One example is LISPDHT [MI08] which is a hybrid push and pull model based on distributed hash tables (DHT) . DHT provides interesting features such as self-configuration, self-maintenance, scalability, and robustness. The cost of operating a mapping service is still quite unknown and [IB07] evaluates the cost for a pull model concluding that the dynamics and query traffic are comparable but smaller than the DNS case, which provides a good indication in scalability terms. However, such system implies storing and distributing the content to routers, thus we should expect non-negligible traffic to locate an identifier in response to a query. In addition, there is always a trade-off between a router's cache and the needed bandwidth to perform queries to a mapping server. Finding a balance between both is not a trivial matter. As a final remark, RLOCs and mapping servers need to be coherently updated, otherwise routing coherence between end-hosts is not guaranteed.

Another similar approach to reduce the routing table size is to route packets using AS labels. This decision is controversial with some opinions against it and others in favour [SCE+05]. It has the advantage of separating routing from addressing, leaving the routing unchanged, regardless of the type of addressing scheme used (IPv4 or IPv6). The drawback is that a mapping service must also exist between addresses/identifiers and ASes labels much like in the LISP case. HLP [SCE+05] is an example of a routing protocol that uses

AS numbers to route packets as a scaling mechanism.

The approach followed in [MWZ07] considers that the ASes are organized in a hierarchical form and the address space is separated between end sites (stub ASes) and transit ASes. The claim is that ISPs are far less than end sites and change a lot less as well. The two separate spaces for the addresses are the globally routable addresses (GRA) , and the globally deliverable addresses (GDA). A mapping service, that is not specified, is needed to map the GDA addresses to the correspondent GRA. Routing is performed by encapsulating packets in a tunnel that routes the packets from the source ISP to the destination ISP in the GRA space. The main scalability gains come from the fact that end sites do not impact the routing tables in the GRA space. Table 3.2.1 summarizes the most relevant aspects of the various approaches just described.

Literature summary - Scalability

Reference	**Approach**	**Strengths**	**Weaknesses**
[dLQB06]	IPv6 with multiple provider dependent address per host	Supports address aggregation with multi-homing; increases the number of available paths	Only works for IPV6; depends on multiple provider-dependent addresses
[KcFB07], [Nor09]	Compact routing	Sub-linear routing table growth	Does not deal well with network dynamism
[D.07][MI08], [IB07], [Nor09], [QIdLB07]	Locator Identifier separation	Better scaling due to separation between identifiers and locators; provides more path diversity	Relies on tunnelling for forwarding and needs a mapping service
[MWZ07]	Scalable routing via address space separation	Routing table growth contained in the core	Relies on mapping and tunnelling
[SCE+05]	Routing using AS numbers	Constant one time routing table size reduction	Needs a mapping service; one time only reduction

3.3 Failure management

In BGP a single failure can force all BGP routers to exchange large amounts of BGP advertisements. This process, known as *path exploration* [HRA10], has the objective of exploring alternative paths to reach the failure affected destinations. The traffic produced

is usually called *churn*.

An increase in the amount of *churn* can impair scalability since routers might reach a state where they can no longer deal with the routing dynamics. The size of the routing table has a direct impact in the *churn* rate since more prefixes mean more possible routing events. The evolution of the topology can cause an increase in *churn* that does not scale [EKD10]. A second problem is the convergence time of the routing protocol after a routing event. This convergence time is quite high for BGP even without considering scaling problems.

A common practice in BGP is to use the minimum routing advertising interval (MRAI) timer to avoid update storms. However, poor configuration of this timer can actually increase the convergence time of BGP. In [GP01] it is shown that important updates might be lost due to the minimum advertisement interval. They show that an optimum MRAI value exists depending on the specific network topology. Obviously this is very hard to find in practice since it varies from network from network. Another commonly used practice is route dampening, that is used to avoid BGP update storms by ignoring frequent changing flapping routes. This practice reduces *churn* but, as a side effect, it increases the convergence time of BGP [MGVK02]. Another simple approach to failure management is ghost flushing [BBAS03]. It is a technique in which a withdraw message (that is not subjected to MRAI) is sent immediately to all neighbours whenever a node receives an update with a worse ASPATH (this means that the previous UPDATE that was sent was erroneous and must be flushed) for a given destination and MRAI has not elapsed yet. The idea is to speed convergence by accelerating the path exploration phase. However, this approach actually increases churn thus having a negative impact in scalability. We can see at this point that failure management also disturbs the normal functioning of the network. When a single path is available traffic is interrupted until new routes are established, increasing the need for a fast re-convergence time.

3.3.1 Literature

There are some proposals regarding limiting the path exploration problem by means of root cause identification in BGP updates. In these approaches the origin of the failure is

added to the UPDATE messages whenever a router detects a failure in one of its links. This root cause is represented by a sequence number that indicates a route change in a given router [PAMZ05]. With this information the routers can ignore invalid routes being learned that contain the origin router in the ASPATH but have a smaller sequence number (for that router). In the EPIC approach [CDZK05] a similar effect is obtained by adding a forward edge sequence number to the ASPATH that distinguishes the exit point of an AS and so contains information about the real path that an UPDATE message has travelled. In case of a failure the edge sequence number in the withdraw message identifies the origin of the fault and all UPDATES that contain this edge sequence number are ignored.

Major problems regarding root cause notification methods are the difficulties in adding information to BGP in the large scale needed to make them effective. It can also impact the routing table size since route aggregation would have to be reduced, in order to have more detail on reachability information to prevent two different faults of producing the same BGP UPDATE message [CDZK05].

Other options, like the one proposed in [FMM+04], try to correlate the data observed in strategic points along with the time of occurrence and prefixes of the UPDATE messages to pinpoint the source of a failure. However, such approaches do not reduce the BGP convergence time since they are performed off-line.

In [BFF07] a fast re-route MPLS technique is used to solve short duration failures in inter-domain without starting a BGP re-convergence process. This is achieved by pre-computing an alternative route that is used to set up a protected MPLS tunnel between the ASes. A BGP re-convergence process is only started if the failure persists over a certain time, thus avoiding starting a BGP convergence process for short duration failures. If the link comes up again before this time elapses, the protection tunnel can be deactivated and BGP does not even know that a failure occurred.

The establishment of the protection tunnels can be manual although an auto discovery mechanism is also described. For this later case, BGP has to be extended with a new type of BGP routes called protection routes that contain information about eBGP sessions and are distributed inside de AS through iBGP. This is necessary because all BGP routing tables of the egress routers must contain all the exit routes available. The primary egress

router can then choose an alternative path and establish a MPLS protection tunnel. This approach relies on ASes advertising prefixes through multiple links so that the needed routes for the protection tunnels are available. Of course this has an impact on the scalability of the routing tables size since it impairs address aggregation as described in section 3.2. Moreover, out of band tunnel set up has a processing cost.

Another similar solution is the one presented in [Mag07]. In this case BGP is enhanced to distribute a strategically chosen set of fail-over paths besides the best route. The idea is to have a new route immediately available in a failure so that transient faults (and the subsequent BGP convergence process) do not cause loss of traffic. The concern is not to speed the BGP convergence time but instead to protect data forwarding during that time. As a drawback we have again the scaling problem of multipath advertisement in BGP.

A different approach is to avoid the BGP convergence process altogether. In [LCR+07] this approach is followed. The idea is based on a distinction between transient outages of links and long term link failures, like decommissioned links and planned outages. The network map in each node does not need to have all the transient failures information and only long term changes have to be known in all nodes.

The system works by using a special type of packet called failure carrying packet (FCP). This packet contains failed links that the packet encountered in its path, and this information allows route re-computation in each hop by finding the new best-path using the network map in the routers minus the failed links carried in the packet. The whole idea is that a router only needs to know the set of failed links, in addition to the network map it holds, to compute a path towards a destination. This implies that all nodes have consistent network maps which is not the BGP scenario, since for a path vector protocol such maps are not available. To overcome this, a mechanism where source routing is used is presented, to extend the use of FCP to the BGP scenario. The concept consists on having a packet containing not only the failed links but also the entire route inserted by the first router of the path. Another extension deals with AS policies that might be broken if we used source FCP with no policy concerns. The solution is to treat a policy violation as a link failure. The authors claim that no transient loops occur during BGP convergence after a link failure. According to them, in the worst case, the packet is only forwarded

to a point where there are no more paths available (based on the failed links list) and is dropped. The main deployment problem of such an approach would be the insertion of the AS path on every IP packet, which has some overhead. These issues are discussed in the paper. No simulations or other analysis to try to measure the efficiency of such an approach in the inter-domain case are presented.

The HLP [SCE+05] proposal introduces link state routing in the inter-domain scenario as a means of improving the convergence time after failures. In order to make it possible to use link state routing the network is divided into hierarchies. The hierarchies are defined based on the provider-customer relationships between the ASes. The link state routing is modified inside a hierarchy so that no provider routes are forwarded to other providers or peers in order to comply to one of the most common polices used in the Internet [CR05]. Routing between hierarchies is done using a path vector protocol similar to BGP. The idea is to get the improved converge times of link state routing and obtain fault isolation within the hierarchies, reducing *churn*. However, the policy model is quite restrictive and also the architecture does not respond well to increases of the amount of multihoming and peering agreements in the network. If these links start to exist it will cause ASes to belong to several hierarchies causing scalability problems, reducing the achieved fault isolation and reduction of *churn*.

In [MWZ07] the separation of addresses in GDA and in GRA also improves to some extent the *churn* rate since events in the GDA space (customer networks) have no impact in the GRA space. This improves stability in the core and provides better convergence and improved churn rate than in a global visibility architecture like BGP. Table 3.1 summarizes the most relevant aspects of the various approaches described.

3.4 Multipath routing

In its current release BGP is single path. This means that only the best path is selected and advertised to neighbours. It also means that possible alternative paths are not known in other routers which poses limitations in terms of traffic engineering, both because some networks that might be preferred are not visible and because traffic control can only be made at the prefix level without any finer grain control. Some vendors have

Table 3.1: Literature summary - Failure handling

Reference	Approach	Strengths	Weaknesses
[MWZ07]	Address space separation	Fault isolation between different address spaces	Relies on mapping and tunnelling
[SCE+05]	Using link state routing within provider customer hierarchies	Improved failure handling using link state routing; fault isolation within hierarchies	ASes can belong to to many hierarchies; relies on a hierarchical structure; needs a mapping service
[PAMZ05], [CDZK05]	Route cause notification methods	Accelerates re-convergence by reducing path exploration	Scalability implications; difficulty in adding the information to messages
[BFF07]	Inter-domain MPLS fast re-route	Fast re-route avoiding BGP convergence process	Needs tunnelling; impacts on scalability; reduces aggregation; needs out of band tunnel set up
[Mag07]	pre-computation of fail safe paths	Avoids re-convergence by having a set of pre-computed paths ready to be used	impacts on scalability
[LCR+07]	Failure carrying packets	Avoids the re-convergence process by appending a list of failed links to packets	Needs consistent network maps or source routing

multipath extensions, and there are some efforts to extend BGP so that it can advertise multiple routes. In [SRW11] an extension to BGP is proposed allowing routers to advertise multiple routes. However, it is still very unclear how to improve BGP with multipath routing capabilities without compromising its scalability, since advertising more routes implies having more entities in the router tables and also more UPDATE messages being exchanged.

3.4.1 Literature

In [HR08] the authors present a survey of the several proposals for multipath routing in the Internet. The main difficulties in the inter-domain scenario are scalability and coordination. Scalability is an issue both at the control plane and the forwarding plane. At the control plane more paths need to be calculated and distributed thus needing more

processing power and extra bandwidth for routing messages. At the forwarding level, forwarding on multiple paths produces more entries in the forwarding tables. Moreover, it usually requires an extra header or label in order to avoid forwarding loops or to assure that a given path is followed.

The most used forwarding mechanism in the Internet is destination based hop-by-hop forwarding. Forwarding on multiple alternative paths using such a mechanism is impaired by two problems:

- Destination based forwarding decisions do not allow a source node to chose the entire path towards the destination; it only chooses the next hop. If another path (different from the one preferred by the source) is available at the next hop there is no way for the source to avoid it from being used since the decision is made by the next hop.

- Allowing several paths increases the possibility of forwarding loops. If only the shortest path is available the forwarding decisions are strictly monotonically decreasing, the number of hops towards the destination and no forwarding loops occur[1]. Allowing other paths without changing the overall forwarding paradigm can break this decreasing property and forwarding loops can occur.

The limits and trade-offs on the amount of different paths that can be used at each node for destination based hop by hop forwarding are discussed in chapter 6. In any case, it is only possible to assure that a specific end to end path is used, if some alternative form of forwarding is used. There are two main alternative forwarding schemes: tunnelling and explicit routing. Both forms imply the use of a second packet header or a label.

Obtaining more paths is also an issue. LISP is one of the solutions that provides path diversity but with quite specific characteristics. Its advantages are evaluated in [QIdLB07] based on a proposal for an incremental implementation of the locator/identifier separation through an IP over IP tunnelling protocol. The mapping function can return more than one locator for each identifier thus providing a certain flavour of multiple paths. The AS of the origin can then choose amongst the various locators based on some traffic engineering criteria. Other proposals also increase the number of known paths but are not concerned with the simultaneous use of more than one path for forwarding [BFF07],[Mag07].

[1] In chapter 4 the formal foundation for this claim is presented

A true multipath routing approach is MIRO described in [XR06]. In MIRO the routers continue to use BGP for the default route but arbitrary pairs of domains can negotiate the simultaneous use of additional paths. In the negotiation process new routes can be advertised by one AS to the other based on policies. The traffic on the alternate route is directed using tunnels, and it can happen that some part of the traffic is forwarded based on the destination prefix while the other part uses the tunnel.

NIRA [YCB07] is a proposal for an inter-domain routing architecture in which each host has the knowledge of a partial graph of the network that comprises the ASes to which it is connected to until the core level (ASes with no providers are considered as belonging to the core level). This partial graph contains all connections of these ASes. With this information routes are chosen by the hosts and packets are forwarded by routers along these routes using encapsulation. As there is the knowledge of this part of the network multiple possible routes can be chosen by the host. However, the system needs a mapping service to map the end hosts to the core ASes since hosts only know the map to the core. NIRA also requires addressing to be hierarchical and the use of IPv6. The architecture is also heavily based on the provider-customer hierarchy and does not deal well with peering links. Table 3.2 summarizes the main strengths and weaknesses of the reviewed literature.

3.5 Traffic engineering

Traffic engineering (TE) can be seen generally as the distribution of traffic to suit a specific goal. It has evolved with the need to efficiently use network resources and in close relation with the need to provide Quality of Service (QoS). At the routing level, it can be seen as a routing optimization problem. The complexity of the routing optimization problem depends essentially on the amount of traffic control provided by the routing platform and the available information about the network and traffic. The characteristics of the inter-domain scenario are quite defined: the amount of traffic control is usually small, the routing is based on a single path system (BGP), and the current TE works mainly on the choice of a specific path for each destination with techniques based on manipulating the route decision process by setting the appropriate attributes in the announced routes [QUP+03]. Note that information about network conditions can be hard to obtain due to

Table 3.2: Literature summary - Multipath

Reference	Approach	Strengths	Weaknesses
[D.07][MI08], [IB07], [Nor09], [QIdLB07]	Locator Identifier Separation with mapping service	Increases the number of available paths by mapping an identifier to more than one locator	Relies on tunnelling for forwarding and needs a mapping service
[BFF07]	Inter-domain MPLS fast re-route	Increases the number of available paths since ASes announce alternatives	Needs tunnelling; impacts on scalability reducing aggregation; needs out of band tunnel set up
[Mag07]	pre-computation of fail safe paths	Increases the number of available paths	Impacts on scalabilty
[XR06]	Negotiation of alternative paths between ASes	Provides multipath routing forwarding on multiple paths simultaneously	Needs cooperation between ASes and tunnelling for forwarding in the extra paths
[YCB07]	Separating the path in two segments Source - Core and Core - Destination	Provides multiple routes since destinations can be reached by more than one core AS	Needs a mapping system; needs partial knowledge of the network map; requires hierarchical IPv6 addressing and forwarding using packet encapsulation

the network scale and the distributed nature of the network control. With such complex traffic control and scarce network information, inter-domain TE is usually limited to local objectives for each individual AS such as choosing the entry (inbound) and exit (outbound) points of traffic. In this context different ASes have different needs. For example, an AS hosting a popular content provider needs to control outbound traffic, while an AS having multiple application subscribers wants to control its inbound traffic. A transit domain moving traffic between ASes might need both. Although different in purpose the two types cannot be seen as independent since one impacts the other [FPMB07].

Outbound traffic control is more easily managed than inbound. ASes control the decision process by tuning the border routers to use the appropriate attributes, usually using the LOCAL-PREF attribute. Also, they can tune the IGP so that the proper border router is chosen for each destination [QUP+03]. However, this control is always limited

by the available routes received from neighbours since BGP is single path.

Inbound traffic control is a more complicated subject since it involves influencing the BGP decision process on other ASes. Several techniques are used [QUP+03] such as selective announcements, prefix de-aggregation, AS-Path prepending, MED, and redistribution communities. These techniques are rudimentary and pose a lot of constraints on the TE problem. There are side effects like the increase on the number of BGP advertisements caused by prefix de-aggregation that has an impact on the default free zone (DFZ) routing table size [BGT04]. Also, the interaction between attributes in BGP might not lead to the expected results. The optional community attribute is also commonly used for TE purposes [QTUB04]. It works by overloading the meaning of the route announcements. The precise meaning is set between neighbours as well as the actions to perform. The main drawback [QTUB04] is the absence of a structured definition for the values and the lack of a standard method of advertising them to others.

The difficulty in achieving inter-domain TE is rooted in the fact that traffic control is deeply connected to route dissemination. This poses problems, since routing decisions and configurations are independent in each AS, and changes in one AS affect the decisions in other ASes and can propagate in a cascade causing unpredicted instabilities. These instabilities can happen because the convergence to a set of stable policy compliant paths in finite time is not a sure fact in BGP as it will be pointed in the next section. This is even worse when changes are performed without coordination or lack of some knowledge of the network [YXW+05].

3.5.1 Literature

Inter-domain TE research has been focused mainly in providing enhanced control for the traditional outbound and inbound TE problems. Let's us start with the outbound TE research. To allow a finer control over outgoing traffic, explicit routing using inter-domain MPLS tunnels has been proposed [DdOV07]. ASes can set the tunnels manually, based on the knowledge of the inter-domain links, or automatically based on traffic measurements. With this increased control several methodologies have been proposed that define local optimization problems and propose treatable heuristic algorithms to solve them. One

example is [Uhl04] that proposes an algorithm to minimize expenses and maximize load balancing in stub ASes. More complex schemes, like MESCAL [HFP+05], aim at finding the optimal egress points that comply with a specific end-to-end bandwidth guarantee. MESCAL implies coordination between all ASes involved based on the definition of Service Level Agreements (SLAs) between them. MESCAL uses both off-line and on-line TE techniques. The off-line techniques make use of traffic forecast matrices. A centralized routing server in the AS deals with on-line TE. Lee et al.[LZN04] describes an on-line approach for stub ASes that uses traffic measurements as an input for egress selection.

Some optimization algorithms for outbound traffic control exist. Wang et al.[WCC+05], for example, calculates the amount of AS prepending for each prefix to achieve an objective but it needs to have information on maximum link loads and knowledge of the topology. In [SAM07] a cooperative scheme using game theory and non-linear programming is used to solve a peering problem between two domains that agree on having a peering relationship. It determines how many peering points are needed and how ingress and egress points should be used by both ASes in order to maximize a given utility function. In this case signalling between the cooperating ASes is needed. In [QB05] inbound TE is performed by using IP tunnelling to explicitly route between source and destination domains.

In summary, these solutions are mostly heuristic or try to mix optimization problems with heuristic algorithms to optimize single AS inbound and outbound traffic. To deal with the limitations caused by the coupling between traffic control and routing dissemination there are two types of solutions (some approaches combine both): tunnelling and cooperation. Tunnelling (using MPLS or other form) provides explicit end-to-end routes and traffic control becomes natural. However, tunnelling requires cooperation and signalling. Also, maintaining end-to-end inter-domain tunnels without affecting scalability in a network as large as the Internet is still an open problem. Cooperation between ASes uses network information like topology, traffic demands and current conditions to try to find the correct route manipulations so that changing the route dissemination does not cause instabilities. A trade-off between simplicity and optimality is clearly visible: better TE depends on better traffic control and this depends on new and more complex routing features, and on signalling and cooperation between ASes. Table 3.3 summarizes the

reviewed literature.

Table 3.3: Literature summary - Traffic engineering

Reference	Approach	Strengths	Weaknesses
[DdOV07]	Explicit routing using interdomain MPLS tunnels	Increases the control of the outbound traffic	Uses tunnelling; needs information on traffic and /or the existent links between the ASes
[Uhl04]	local optimization problem solved with heuristics	Applied to cost minimization and load balancing maximizing problems	Only deals with a local problem; it is performed off-line
[HFP+05]	SLAs defined between coordinated ASes	Optimizes egress points using	Needs coordination, a centralized routing server and traffic forecast matrices
[WCC+05]	Off-line AS pre-pending optimization	optimizes AS pre-pending	Relies on AS pre-pending; needs information on link loads and topology
[SAM07]	Off-line optimization of peering links using game theory and non-linear programming	Determines how two ASes should peer maximizing an utility function	Needs signalling and co-operation between ASes
[QB05]	Explicit routing using interdomain MPLS tunnels for inbound TE	Improves control over inbound traffic	Uses tunnelling; needs information on traffic and /or the existent links between the ASes

3.6 Expressiveness and safety of policies

BGP allows absolute freedom in the application of the routing policy. Its attribute based decision process allows a great number of possible tweaks in defining which path is considered the best path. A filtering process allows extra control on which routes are exported to neighbours. The end result is a complex system where the final routing state depends on the result of several independently applied routing policies and mechanisms. On one hand BGP has sufficiently flexibility to construct intricate internal policies but on the other hand it does not have capabilities to attach information to a route in order to share the information through the network. This means that without any form of coordination

or restriction in the application of policies by each individual AS, it is not assured in the current architecture that the system can converge to a stable routing state in finite time. A simple example of how a stable routing state can not be achieved is the bad gadget example in figure 3.1.

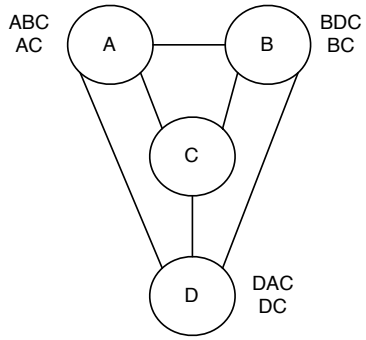

Figure 3.1: Bad gadget

In this example we have a graph of the network and the permitted paths for each of the ASes ordered by their preference order. Let's assume that due to policy reasons other routes like for example ADC for AS A are not permitted. We can see that each AS prefers the indirect path to the central node instead of the direct one. In this case the network never reaches a stable set of paths. The process is the following: when the direct routes (BC;AC;DC) are known, the indirect routes (ABC;DAC;BDC) become available and are advertised to the neighbours. Since they are more preferred they are chosen and the direct ones are withdrawn. At this point the indirect routes are no longer available and are withdrawn, causing the direct routes to be advertised again and the process never stops.

Problems such as the bad gadget example are difficult to detect because they involve policies of different ASes. The problem is the conflicting nature of the objectives of policy routing protocols. There are three main goals:

- Robustness. That is the ability to converge to a unique stable set of paths in the network even if we remove some of the edges or nodes (stable routing behaviour even in case of failures).

- Expressiveness; is the ability of the protocol to model as many different network

configurations as possible. Note that in order to increase robustness, the expressiveness of the protocol might decrease because restrictions to the possible network configurations might be needed [JR05].

- Autonomy is the amount of coordination needed to obtain robustness in a given protocol; the ideal would be that the nodes policies could be written independently, from the policies of other nodes.

There are inherent trade-offs that must be taken in consideration between these three aspects. In the BGP case there are no restrictions on the possible policies (high flexibility) and nodes are assumed to be completely autonomous (high autonomy). The result is that robustness is not assured.

3.6.1 Literature

In [MC05] a series of possible solutions for BGP divergence (absence of robustness) are presented. There are three types of approaches.

The first approach is static routing policies checking. The ASes share their routing policies and the mechanism proposed is collecting them at a central database. This implies the definition of a common language to describe policies and also that the ASes are willing to disclose and share their local policies. An open point is how to decide whether a set of policies leads to divergence or not. It has been shown in [GSW02] that this is a NP-hard problem. Anyway, this first approach reduces node autonomy in the routing protocol.

The second approach is runtime policy analysis. In this category there are several solutions.

The first solution is collecting path histories [GW00]. For each path an AS maintains a history of path changes that leads to the current path being adopted. With this solution it is possible to detect cycles in the paths adopted. In the example of figure 3.1 above there is a repetition of the selected paths in the history and therefore the bad gadget anomaly would be detected.

The occurrence of repetition of a path in the history of the chosen paths is a necessary but not sufficient condition for BGP divergence. When it happens in a divergent situation it allows a more easy pinpoint of the error. The divergence situation might solved by

removing the repeated path from the set of paths allowed by the AS. This method has some drawbacks. Two of them are: the amount of memory needed to store the path histories can be several times the amount needed to store the normal ASPATH; and path histories can partially reveal the routing policies of the ASes.

The third approach is to have path choice restriction via diffusing computations [CGM03]. This approach is based on the observation that if an AS only changes its preferred path to a new path with an equal or greater LOCAL_PREF value, the system cannot diverge. It means that the LOCAL_PREF value of the paths chosen in all ASes must monotonically increase. The proposed solution enforces this through coordination between the ASes using diffusing computation along the routing tree [GLA93]. The system avoids divergence but as it requires AS cooperation and the diffusion computation along the routing tree it adds message overhead.

A previous work [MC04] by the authors of [MC05] is also based on the same observation that if the preference in the current path in a AS monotonically increases, the divergence will not occur. To detect divergence they propose that each AS advertises a metric value that increases every time a new UPDATE causes an AS to change to a less preferred rank path. In case of divergence the metric value will continue to increase periodically. When this metric reaches a certain value it means that a cyclic divergent behaviour is occurring and the AS receiving the UPDATE does not accept the new path. In this solution there is less overhead since only a single integer value has to be added to the UPDATE messages. The policy is only restricted when the threshold is reached. This procedure has the drawback that convergence will be slower since an erratic divergence causing misconfiguration will only be detected when the threshold is reached. Since rank decreasing is not a sufficient condition for divergence there can still be situations when the policy restriction was not necessary and even so the solution detects a divergence. Table 3.4 presents a summary of the literature described above.

3.7 Summary

This chapter has shown some of the most important challenges that have been brought to the inter-domain level. The initial simple task of routing is already very far from the

Table 3.4: Literature summary - Expressiveness and safety of policies

Reference	Approach	Strengths	Weaknesses
[GW00]	Path histories	Detects divergent configurations through cycles in the histories	Amount of memory needed for path histories; a cycle in the histories is not a sufficient condition for divergence
[CGM03]	Policy restriction (increasing LOCAL_PREF in selected paths) via diffusing computations	Assures convergence to a stable routing state	Needs coordination between ASes to enforce the policies; the diffusion computations along the routing tree add message overhead.

current use of the network, even though BGP is being called to perform all tasks.

With the evolution of the network the highly tunable nature of BGP has lead to its usage for traffic engineering purposes like defining backup paths or trying to control the inbound or outbound traffic of an AS. It ends up in a scenario where very flexible policies can be applied individually by each AS according to its particular objectives. All these endeavours resulted in a complex system based on the manipulation of routes at prefix granularity that exhibits several problems. Figure 3.2 illustrates the relationships between the aspects covered in this chapter for the current network architecture, and illustrates the difficulties in solving each of them individually without causing possible negative influences on the others.

The need for Traffic Engineering (TE) leads to an increasing level of multihoming in the network, both for reliability and for load balancing purposes. TE was also the reason for the increased number of peering links on the network. It is a fact that in terms of the current Internet technology in order to control traffic we need to influence routing. This implies either prefix de-aggregation, or changing the destination based hop-by-hop forwarding method. Both situations have scalability problems: prefix de-aggregation implies more entries, and changing the forwarding method implies signalling and more routing state. Another implication introduced by the application of TE techniques is policy safeness. Many of the used prefix manipulations have outcomes difficult to predict when they are propagated through the network. This may lead to a non-stable routing state, that

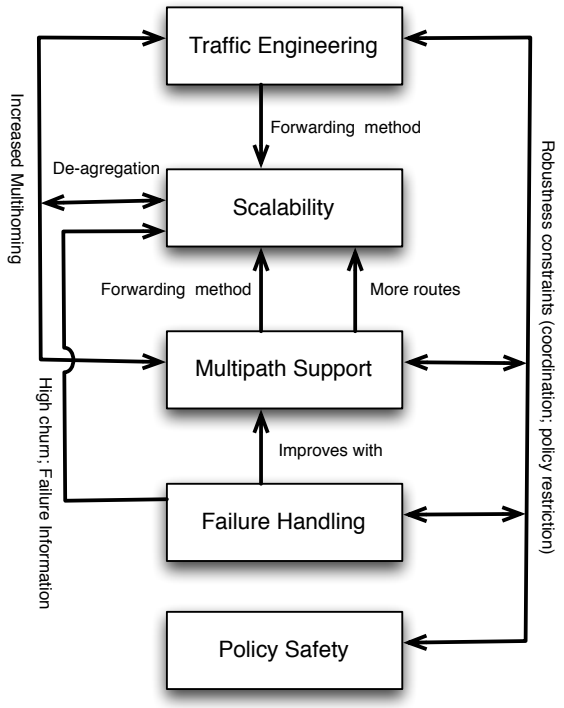

Figure 3.2: Inter-domain routing architecture relationship between aspects

can even be more unpredictable in the presence of failures. Adding support for multipath can improve the ability to perform better TE. The alternative paths can also be used to achieve faster convergence times after failures. However, it is still not clear how to increase the number of paths without increasing the already present scalability problems in BGP. Most of the available multipath solutions rely on the use of tunnelling or other forms of encapsulated forwarding, but inevitably more routes have to be stored in the routing tables. Proposals to accelerate re-convergence after a failure that rely on information to pin point the cause of failures also impairs scalability since they add overhead in UPDATE messages and reduce aggregation. Finally, when using BGP, robustness is not assured unless some restrictions on policy are enforced. This aspect is worsened by failures, multipath support and some of the proposed traffic engineering techniques.

Some of the just presented proposals are alternative routing architectures that change some fundamental aspects of the current architecture. Three main aspects are easily identifiable:

- Changing the existing flat address space. Approaches range from simply routing

via ASes numbers like in HLP [SCE+05] to separating address space [MWZ07] or separation between identifiers and locators like in LISP[D.07]. Changing the address space provides a one time constant reduction factor in the number of routable objects but it also implies the existence of a mapping service to translate between the different address granularities. Much of the scalability pressure is transferred to that mapping service. Having a mapping service can open interesting possibilities.It can be a means to add the possibility to find multiple paths or to support more advanced features like host mobility. Another possible advantage of a separation in the address space is failure isolation.

- Changing the routing paradigm to avoid slow convergence after failures. HLP [SCE+05] tries to bring link state routing to the inter-domain scenario. In [BFF07] MPLS fast re-route is brought to inter AS links, and in [LCR+07] it is proposed to use a constraint shortest path algorithm using failure information provided by a signalling packet. In [KcFB07] and [Nor09] compact routing is proposed. In NIRA [YCB07] hosts choose routes based on a partial map to the core with the aid of a mapping system.

- Restricting or defining sets of policies to be used to assure correctness. HLP [SCE+05] and NIRA [YCB07] are examples of policy restriction. A related issue is the cooperation between ASes to maintain coherent policies [QB05],[SAM07], [XR06] are all approaches that require cooperation between ASes.

At this point the open problems can be identified more clearly. There is a need for a routing architecture that is based on semantically rich policies. This means that at the inter-domain level the best path is defined by a set of policies that are related to the individual objectives of each node (AS) of the network. These objectives are related to business and traffic engineering objectives and are known only by each individual AS. So far so good, but this richness of policies combined with the lack of any coordination between nodes or knowledge of the applied policies on other nodes leads to a routing architecture very prone to unstable routing states. Some of this is caused by the fact that everything in BGP is performed by tweaking routing attributes. It leads to a very complex system

where some of the interactions can be unpredictable. Some kind of separation should exist between routing and other functions like TE. The control of where the traffic flows should not affect the routing state and should be performed on top of a stable set of routes. Achieving such separation in a single path routing system seems very hard. The ideal would be to have a stable set of paths that can have multiple paths per destination on top of which we might distribute traffic according to the objectives.

The availability of multiple paths also allows a better handling of failures: alternative paths can be immediately used and the re-convergence time of routing to the updated routing state becomes less critical. One drawback of multiple paths is the increase of the robustness problems caused by independent routing decisions and usually destination based hop by hop forwarding is replaced with source routing, tunnelling or other form of packet encapsulation. Unfortunately these replacements are difficult to implement at the inter-domain level. Finally, it is difficult to assure that a stable routing state is always achieved. Achieving this stability implies either coordination between nodes or some restriction on the policies and routing decisions.

In short, we need a multipath policy based routing architecture. In the context of the inter-domain scenario, this should be achieved with minimum need of coordination between nodes and ideally with independent routing and forwarding decisions. In order to understand how this can be achieved we need to have a formal model of routing that allows us to express a given routing protocol and verify its correctness formally. In the next chapter a series of relevant approaches to formally model path vector protocols like BGP are presented. Then, they are related with a generic algebraic and graph theoretical approach that is used in this thesis.

Chapter 4

Theoretical models of routing

4.1 Introduction

One of the issues discussed in chapter 3 was the expressiveness and safety of policies (section 3.6). This issue is transversal to all other issues since in fact it deals with the *correctness* of the routing protocol. A routing protocol is said to be *correct* if it reaches a stable routing state in finite time. This means that a set of stable routes is calculated in finite time. A second aspect of *correctness* is the absence of forwarding loops, and in this thesis we are focused on simple destination based hop-by-hop forwarding. Therefore, whenever it is referred that a routing protocol is *correct* it means that it complies with the following definition.

Definition 4.1. A routing protocol is *correct* if and only if

- It converges to a routing solution, meaning that it calculates a stable set of routes in finite time.
- It forwards packets without causing loops in *destination based hop-by-hop* forwarding.

Understanding under which conditions a routing protocol can operate correctly is fundamental to designing a multipath policy based routing protocol that suits the inter-domain routing scenario. Ideally, we should have some sort of theoretical model for routing that would allow us to verify if a given protocol is *correct*.

Routing can be divided in two main components: *policy* and *algorithm*. The policy provides:

1. A meaning for the attributes of a link.

2. A definition on how the attributes of a route are obtained from the sequence of links.

3. Rules of preference amongst valid routes.

The algorithm refers to:

1. How the network map is obtained.

2. How the routes are calculated.

3. How routes are compared and selected according to what is defined by the policy part.

For example, in a traditional shortest-path routing protocol like OSPF or IS-IS, the policy part consists on: assigning a numeric link metric to all links; defining the route value as the sum of the metric of all links; and defining that the best path between a source and a destination is the one with the shortest path value.

The algorithm part of such protocols is the use of link state packets to discover the network map, and the use of Dijkstra's algorithm to calculate the shortest paths.

Routing protocols have evolved, and are usually presented, in an algorithmic view turning the task of separating the policy part from the algorithm a challenge. In the rest of this chapter two formal approaches developed to model BGP-like inter-domain routing protocols are presented. After this, a pure algebraic and graph theory approach is deduced, starting by briefly presenting the origins of algebraic routing modelling and then presenting the latest evolutions in modelling policy rich protocols. Lastly, the convergence conditions for routing protocols (with the appropriate characteristics to solve the problems identified in chapter 3) are then systematized. Some original contributions are made in isolating the key properties for multipath policy routing convergence using destination based hop by hop forwarding. Also some trade-off's between policy freeness and network restrictions are identified when trying to achieve the highest degree of multiple paths while still assuring protocol correctness.

4.2 The Stable Paths Problem (SPP)

A first approach for modelling policy rich routing protocols was to study policies on individual networks. In [GSW02] the stable paths problem (SPP) framework was developed. It is a formalization of BGP based on a simple graph-theoretical approach. It defines an asynchronous protocol of the Bellman-Ford class for computing routing solutions that represent a stable set of paths in a Nash equilibrium [1]. An instance of the SPP contains the graph of the network, a set of the permitted paths in each node, and a rank assigned for each path that corresponds to its level of preference for the node. In general terms, SPP works on a graph $G(V, E)$ of the network. For each $v \in V$ it defines a set of paths P^v that contains all permitted paths from the node v to a given origin node v_0. There is also a non negative integer valued function λ^v defined over P^v that represents the rank of preference amongst the paths of the set. If P_a and P_b belong to P^v and if P_b is preferred to P_a then we have $\lambda^v(P_a) < \lambda^v(P_b)$, meaning that the greater the value the more preferred.

The use of SPP leads to several interesting conclusions. The first is that finding the solution for a particular SPP instance is an NP-complete problem. This means that checking if the protocol is *correct* for a particular choice of policies is impractical. However, the authors did find a set of sufficient conditions on network policies for which the protocol is assured to be *correct*. These conditions are based on the absence of a structure called a *dispute wheel*.

Definition 4.2. Consider a sequence of nodes $x_0, x_1, \cdots, x_{k-1}$ and sequences of paths $Q_0, Q_1, \cdots, Q_{k-1}$ and $R_0, R_1, \cdots, R_{k-1}$ (see Figure 4.1) such that for each $0 \leq i \leq k-1$ we have:

- R_i is a path from x_i to x_{i+1}
- $Q_i \in P^{x_i}$.
- The path composed by R_i appended with Q_{i+1}, R_iQ_{i+1}, also belongs to P^{x_i}.

If $\lambda^{x_i}(Q_i) < \lambda^{x_i}(R_iQ_{i+1})$ we have a *dispute wheel*

[1] A Nash equilibrium is a routing state from which no node can improve their path choice given its routes

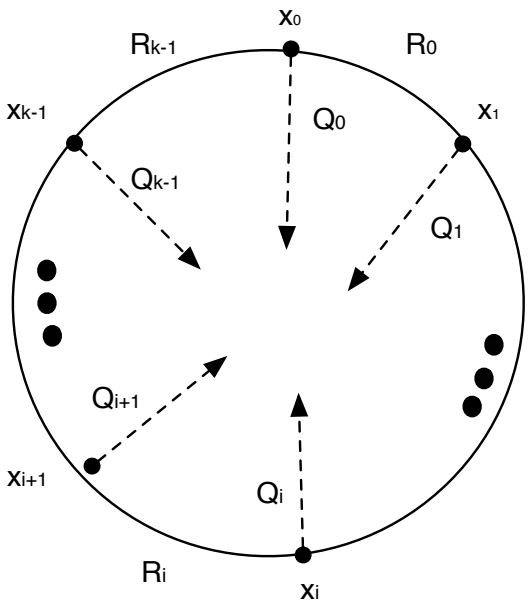

Figure 4.1: Generic dispute wheel

It is easy to see that the bad gadget (see chapter 3 section 3.6) is a particular instance of a *dispute wheel*, since in both cases nodes prefer the path using the next node in the dispute wheel (R_iQ_{i+1}) instead of the direct one (Q_i) when routing. The SPP framework models BGP and a specific set of policies (an SPP instance receives as input the set of permitted paths for all ASes and the preference rank values amongst them). The absence of dispute wheels in the network is a sufficient condition that assures that the protocol convergences to a routing solution. So, we can verify if one protocol converges to a stable routing solution just by verifying the absence of dispute wheels in those particular conditions.

4.3 Path vector algebras

Another theoretical model for BGP-like protocols was proposed in [Sob05]. It follows a more algebraic view and introduces what was called path vector algebras. These algebras come from previous algebraic generalizations of the shortest path routing performed by the same author in [Sob01]. They take the form $(S, \preceq, L, \otimes)$ where:

- S is a set containing the possible values for paths.

- $\preceq$ is a preference order over S.

- L is a set of labels for links.

- $\otimes$ is a binary operation that maps $L \times S$ in to S.

S models the value of a path. This value is obtained by applying the binary operation $\otimes$ to the consecutive links that form the path, where each link has a label $l \in L$ that models the policy applied to it. The advertisement process of BGP, is modelled by this $\otimes$ operation. The $\preceq$ order models the choice of the best path completing the model of the protocol. The algebra is said to be increasing if $\forall a \in S \; a \preceq a \otimes l$ for each $l \in L$, meaning that appending a link to a path always increases its position in the preference order, with the more preferred paths being the ones with the lowest path values in order. This is opposite to the SPP framework where the higher paths in the order were the most preferred ones. The important question is that adding a link to a path causes its preference value to decrease.

The author then proposes an asynchronous distributed algorithm of the Bellman-Ford family that calculates the routing solutions by applying $\otimes$ and $\preceq$. The algorithm itself forbids repetitions of nodes in one path. The author defines the notion of a *non free* cycle (a formal definition is provided later in the chapter) that is closely related with the dispute wheels of the SPP framework. In [RJJR05] a translation between path algebras and SPP is provided. The absence of *non free cycles* in the network is a sufficient condition that assures that the algorithm terminates finding a locally optimal solution. The author also showed that if the algebra is increasing then *non free* cycles never occur in the network.

Recent work has been recasting all this previous models of BGP-like protocols to a pure algebraic setting [Gri08],[Gur09]. The starting point for this algebraic modelling of routing protocols has been the traditional routing models based on algebraic structures known as semirings [GM08], [BT10b]. In this thesis this pure algebraic setting is followed. The next section briefly introduces the semiring routing theory, and then extends it to less restrictive algebraic structures more suited to model multipath policy routing. Convergence conditions are presented both for the calculation of a local optimal routing solution in finite time as well as for loop free destination based hop-by-hop forwarding.

4.4 Algebraic routing

The routing problem is usually modelled in a pure algorithmic way, mixing both the policy and the algorithmic components. If one uses graph theory and linear algebra to model routing, the policy component is modelled algebraically and certain algebraic properties are then studied to provide correctness guarantees when implemented with a certain algorithm.

The network is represented by a directed graph $G(V, E)$. Where V is a set of vertices and E is a set of edges. In order to model the routing problem, we need to model three major aspects: the set of link attributes; the process of obtaining paths by appending links; and finally the process of selecting the paths to be used.

The semiring was the first algebraic structure that was used to model routing protocols [GM08].

Definition 4.3. A semiring is an algebraic structure formed by a set S endowed with two binary operations $\oplus$ and $\otimes$ with the following properties:

- $\oplus$ is commutative : $\forall a, b \in S \; a \oplus b = b \oplus a$
- $\otimes$ distributes over $\oplus$: $\forall a, b, c \in S \; a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$
- $\oplus$ has an identity $\overline{0}$ which is an annihilator for $\otimes$
- $\otimes$ has an identity $\overline{1}$.

The elements of S model link and path attributes, and usually $\overline{0}$ models a non existent or invalid path and $\overline{1}$ models the trivial path, which is the path of one node to itself containing no links.

The $\oplus$ operation models the path choice. In order to model the path choice $\oplus$ should be idempotent, that is $\forall a \in S \; a \oplus a = a$. The $\otimes$ operation models the calculation of paths. It is applied between a link value $l \in S$ and a path value $p \in S$ (starting with the trivial empty path) and results in another path value also from the set S. It expresses how the attributes of a path change with the addition of links.

The $\oplus$'s identity element, $\overline{0}$, provides a natural model for the non-existent or non-valid path, since it is always not selected in $\oplus$ and it is an annihilator of the path composition operation $\otimes$, meaning that any link appended to a non existing or non-valid path will still

result in a non existing or invalid path of value $\overline{0}$. The $\overline{1}$ value, identity element of $\otimes$, models the trivial path since it is the identity of the path composition operation. Although it is not a requirement for the semiring structure, when modelling a routing problem $\overline{1}$ is also an $\oplus$ annihilator since it is selected over any other path value.

This use of semirings to model the routing problem, comes from the fact that such structures are straightforward models for traditional routing models like *shortest path* routing that can be modelled by the following semiring:

$$(\mathbb{N}, min, +)$$

However, to model policy based routing protocols it will be useful to drop some of the properties namely distributivity of $\otimes$ over $\oplus$ [BT10b] [GM08]. So, a more general (less restrictive) algebraic structure formed by a set S endowed with two associative binary operations $\oplus$ and $\otimes$ called bisemigroup is suitable.

Definition 4.4. A bisemigroup is an algebraic structure formed by a set S endowed with two associative binary operations $\oplus$ and $\otimes$ such that we have associativity of $\oplus$

$$\forall a, b \in S \;\; a \oplus (b \oplus c) = (a \oplus b) \oplus c$$

and associativity of $\otimes$

$$\forall a, b \in S \;\; a \otimes (b \otimes c) = (a \otimes b) \otimes c$$

For an idempotent $\oplus$ we can define a *canonical order* $\preceq$ as:

$$b \preceq a \;\; \texttt{if} \;\; a \oplus b = b$$

This can model the path selection operation as a preference order on the possible path values in S. For $\preceq$ we have that for all $a \in S \; \overline{1} \preceq a \preceq \overline{0}$ meaning that the order is bounded between the trivial path (the more preferred) and the invalid or non-existent path (the least preferred). If we substitute $\oplus$ by the derived $\preceq$ order we have a structure called an ordered semigroup [Gri08]:

Definition 4.5. An ordered semigroup is an algebraic structure:

$$(S, \preceq, \otimes)$$

where

$$(S, \oplus, \otimes)$$

is a bisemigroup and the order $\preceq$ is defined by $\oplus$ as

$$b \preceq a \text{ if } a \oplus b = b$$

If in the bisemigroup the set S contains the special elements $\overline{0}$ and $\overline{1}$, then the order $\preceq$ is bounded meaning that for all $a \in S$ $\overline{1} \preceq a \preceq \overline{0}$.

We can use either bisemigroups or ordered semigroups for our model. It is merely a question of considering the path selection as the result of a binary operation over two paths weights or as the result of the relative position of the paths in an order. Throughout this thesis the ordered semigroup is primarily used but in some cases, for readability purposes, we resort to the purely algebraic bisemigroup form.

4.4.1 The routing problem

The routing problem consists on a node finding the best available paths to every other destination in the network. If a solution for the problem exists, then it verifies the first condition of definition 4.1 for protocol correctness.

In general terms the modelling process is the following: a network is modelled by an edge labelled graph $G = (V, E)$ where V is a set of vertices and E is a set of edges. We consider a weight function $w(i, j)$ that maps $E \to S$ (providing values to edges). A path is a sequence of vertices, $P = v_1, v_2 \cdots, v_k$, and its weight is given by the combination of the weights of the edges connecting them, $w(P) = w(v_1, v_2) \otimes w(v_2, v_3) \otimes \cdots \otimes w(v_{k-1}, v_k)$. As discussed above S models the set of possible link and path values, $\overline{0}$ is the non existent or invalid path value, and $\overline{1}$ the trivial path value. $\otimes$ models the path composition operation and the order $\preceq$ the relative preference between the possible path values modelling the path selection process.

The problem can be expressed algebraically by extending the $\otimes$ and $\oplus$ operations to square matrices with the usual definitions [GM08]. A given graph G can be represented by the adjacency matrix:

$$A(i,j) = \begin{cases} w(i,j) & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Consider $P(i,j)$ the set of all paths between i and j. The routing problem is to find for each pair $(i,j) \in V$ the set $O(i,j)$ of the smallest weight paths in $P(i,j)$ according to $\oplus$. This can be expressed as:

$$O(i,j) = \bigoplus_{p \in P(i,j)} w(p)$$

We can find a solution for the routing problem by solving one of two problems:

The left routing problem:

$$L(i,j) = \bigoplus_{q \in V} A(i,q) \otimes L(q,j)$$

corresponds to solving the following matrix equation [GM08]:

$$L = (A \otimes L) \oplus I \tag{4.1}$$

In words, $L(i,j)$ is the smallest weight of paths between nodes i and j given the set of the smallest weight paths of the neighbours q of source i.

The right routing problem:

$$R(i,j) = \bigoplus_{q \in V} R(i,q) \otimes A(q,j)$$

corresponds to the solution of the right matrix equation

$$R = (R \otimes A) \oplus I \tag{4.2}$$

In this case, $R(i,j)$ is the set of the smallest weight paths given the set of the smallest weight paths of the neighbours q of destination j.

The difference between the left and the right local solutions is in the order of the path weight calculation: starting from the destination for the left case, or from the source for the right case. In both cases the solutions of equations (4.1) and (4.2) contain the smallest weights and not the actual paths but we can store the nodes that correspond to these weights during the calculation process and therefore obtain the routing table.

The solutions to equations (4.1) and (4.2) can be found by calculating the closure matrix of the adjacency matrix A denoted by A^* under certain convergence conditions that are expressed by certain algebraic properties of the ordered semigroup [GM08].

4.4.2 Convergence conditions

There are two sufficient but not necessary conditions for the computation of A^* to be possible. The conditions can be expressed either for bisemigroups, in a pure algebraic setting, or for an ordered semigroup.

The first condition is monotonicity for ordered semigroups or distributivity of $\otimes$ over $\oplus$ for bisemigroups. Monotonicity is defined as:

Definition 4.6. An ordered semigroup $(S, \preceq, \otimes)$ is monotone if:

$$\forall a, b, c \in S \; a \preceq b \Rightarrow (c \otimes a) \preceq (c \otimes b)$$

This condition corresponds to distributivity of $\otimes$ over $\oplus$ in a bisemigroup. A bisemigroup has distributivity of $\otimes$ over $\oplus$ if

$$\forall a, b, c \in S \; a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$$

Let's analyse the case where the routing is modelled by an ordered semigroup. The best path is the path that is lowest in the $\preceq$ order. Intuitively, in a monotone ordered semigroup a sub path of a best path is also a best path. This is a sufficient condition for A^* to exist. In this case it is a global optimum solution for the routing problem and it can be obtained either by solving the left or the right routing problems. A^* can be calculated through matrix iteration by consecutively choosing the more $\preceq$ preferred paths between each (i, j) pair of vertices of G. Algebraic iteration methods to find A^* have been related

to the Bellman-Ford and Dijkstra algorithms who are known to compute the solution for respectively the right routing problem equation (4.2) and the left routing problem equation (4.1) [GM08][SG10].

The second sufficient condition, that is less restrictive and does not imply monotonicity is the *increasing* property:

Definition 4.7. An ordered semigroup is *increasing* if

$$\forall a, b \in S \; a \prec a \otimes b$$

This means intuitively that the addition of a link always decreases path preference, increasing its position in the order $\preceq$.

Evidently, a semigroup having both monotonicity and *increasing* properties features the characteristics described previously for monotonicity. An interesting case is to analyse what happens in terms of convergence if we drop monotony. In this case we do not assure that every subpath of a best path is also a best path. However, a solution for the left and for the right equations (4.1) (4.2) can still be found for an *increasing* semigroup. They might not be equal and they will correspond to a local optimum solution where the best routes in one node are dependent on the best routes of the other nodes. Several flavours of proof can be found in [GM08][BT10b][Gri10][Sob05] [Gri10].

In this work we call the non-strict version of definition 4.7 the *non decreasing* property.

Definition 4.8. An ordered semigroup is *non decreasing* if

$$\forall a, b \in S \; a \preceq a \otimes b$$

The *non decreasing* property is not a sufficient condition for convergence.

Independently of the two conditions above, in order to be possible to calculate A^*, the following *non negative* cycle condition must hold in the network graph [GM08][SG10].

Condition 4.1. *Let's denote $w(p)$ as the value in S that corresponds to the result of the path composition operation $\otimes$ over the l_i links of p. A cyclic path c is a path where all nodes are different apart from the first and last nodes which are the same. The network meets the non negative cycle condition if and only if $\overline{1} \preceq w(C)$ $\forall C \in G(V, E)$.*

Condition 4.1 is naturally met for every network labelled with a set S that has lower bound $\overline{1}$. If the condition is not met, travelling around the cycle will continuously produce a more $\preceq$ preferred path and the algorithms do not stop since in each iteration the preference is better.

Another stronger condition can be set on all the entries of the adjacency matrix A: $\overline{1} \preceq A(i,j)\ \forall i,j \in E$. This means that if all edges have a weight equal or worse than $\overline{1}$ the preference will not increase in a monotone semigroup with the addition of any link. This means of course that the previous *non negative* cycle condition will always be met. In this last case, for a monotone ordered semigroup, the Dijkstra's algorithm will also converge to A^* [GM08][SG10].

The *non negative* cycle condition is more relevant in semigroups featuring only the monotony condition, because if they feature the *increasing* condition, it is naturally met.

Multipath convergence conditions

Introducing multipath routing is one of the key issues in inter-domain routing. This implies the use of more than one path to forward packets simultaneously to the destination. Therefore, it is important to derive the convergence conditions in the multipath case, and verify if they are similar to the ones of the single path routing problem.

There are two different but related problems here: the *k-best path* problem and the *multipath* problem. The *k-best path* problem refers to having more than one path between (i,j) with the same weight value, $w \in S$. The *multipath* problem refers to having more than one path with equally preferred weights, $w_1, w_2 \in S$. The *multipath* case implies that $\oplus$ is not selective (selective means that the result must be one of the operands $\forall_{a,b \in S}$ $a \oplus b = a \vee b$), while the *k-best path* case does not. In the multipath case for the order $\preceq$ we can have $a, b \in S$ with $a \neq b$ but $a \simeq b$, meaning that a and b have an equivalent position in the order $\preceq$.

Both problems can be modelled by defining minimal sets of S.

Definition 4.9. A minimal set is a subset $C \subset S$ with cardinality $|C| = n$ such that:

$$C = min_{\preceq}(C) = \{x \in C | \forall y \in C : x \preceq y\}$$

It represents a set of paths with the same preference, meaning that they are either equal $w_1 = \cdots = w_n$ or equivalent $w_1 \simeq \cdots \simeq w_n$. The set of all minimal sets of S is $M(S)$.

An algebra can be defined from an ordered semigroup $(S, \preceq, \otimes)$ in which the minimal sets of S are used and the operators are redefined to work with the minimal sets instead of the individual elements of S.

The operator $\oplus'$ is given by:

$$A \oplus' B = min_{\preceq}(A \cup B) \tag{4.3}$$

and returns the most preferred paths out of the union of A and B.

Likewise, $\otimes'$ is the extension of $\otimes$ to sets of paths:

$$\forall_{a \in S, C \in M(S)} \; a \otimes' C = min_{\preceq}(a \otimes C) \tag{4.4}$$

meaning that $\otimes'$ results in the minimal path weights when $\otimes$ adding a new link a to all elements of the set of minimal paths C for all existent minimal sets $M(S)$.

The resulting multipath algebra is then:

$$(M(S), \oplus', \otimes')$$

[Gur09] proves that if an algebra is monotone the minimal set routing solution can be found and it is a global optimum. Also, it proves that in the absence of monotony, an algebra with a strictly *increasing* $\otimes$ operation is sufficient to assure the convergence to a local optimal routing solution using multiple paths of equal or equivalent preference. In [Cha06] multipath routing is studied using a notion that relates the SPP framework with algebraic routing called policy relation (introduced in [CGG06]). The author finds a similar conclusion to the one in [Gur09]: a multiple path routing solution is possible if the policy relation is anti-reflexive (an anti-reflexive policy relation can be mapped into an *increasing* algebra in a pure algebraic setting).

In summary, the convergence conditions are the same as in the single path case and the algebraic model of the policy part of the protocol has either to be a monotone or-

dered semigroup (definition 4.6) or an ordered semigroup with an *increasing* $\otimes$ operation (definition 4.7).

At this point we have established that the necessary algebraic properties in order for a routing protocol to converge to a solution are the same both for the single and multipath cases. We now examine them more closely, specifically we analyse the impact of relaxing or enforcing strictness.

We already presented in definition 4.8 the non-strict version of the *increasing* property. The monotony condition in 4.6 is non-strict. The strict version of monotony is given by the following definition:

Definition 4.10. An ordered semigroup $(S, \preceq, \otimes)$ is strictly monotone if:

$$\forall_{a,b,c \in S} \; a \prec b \Rightarrow (c \otimes a) \prec (c \otimes b)$$

The sufficient convergence conditions for convergence to a routing solution are: the non-strict version of monotony defined in 4.6 and/or the strict *increasing* property. We analyse the impact of changing these two conditions between their strict and non-strict versions to see if convergence is still possible and in which conditions.

Strictness impact on convergence to a routing solution

We start by studying the differences between monotony as defined in definition 4.6 and its strict version defined in 4.10.

A non-strictly monotone algebra is one of the sufficient conditions for convergence to a routing solution. In such algebras a path travelling around a cycle never decreases in value and therefore it is always increasing or maintaining its position in the $\preceq$ order. For this to be true the only condition is that the network meets the *non negative* cycle condition (condition 4.1).

However, in a non-strict monotone ordered semigroup the following statement might be true:

$$\exists_{a,b \in S} \;\; \text{such that} \;\; a \preceq b \;\; \text{and} \;\; (c \otimes a) \simeq (c \otimes b)$$

This means that preference can be maintained with the addition of links to a path. The consequence of this is that we can have cycles in the network where an infinite number of paths (consecutively passing around the cycle) all have the minimal weight. Although we converge to the minimal global optimum values (because they never increase), we do not obtain all of them in finite time. To say the least, this poses a practical problem when implementing a protocol modelled by such an algebra.

This means that although non-strict monotony is sufficient to ensure convergence, in order to assure finite time convergence (and therefore a practical implementation of the protocol) the model has to be a strictly monotone ordered semigroup.

Let's look now at the case where the model has a strictly *increasing* $\otimes$ operation as defined in definition 4.7. We have already established that this assures convergence to a routing solution. In this case we have finite time convergence, because in a strictly *increasing* ordered semigroup adding a link always increases the path weight and therefore decreases preference. This means that preference around cycles always decreases, and we converge to a solution in finite time.

In summary, considering convergence to a stable routing solution in finite time we need a protocol modelled by an algebraic structure of one of the following two types:

1. A strictly monotone ordered semigroup according to definition 4.10.

2. An ordered semigroup with a strictly *increasing* $\otimes$ operation according to condition 4.7.

Once again this is valid for both single path and multipath routing. This means that only with one of the two strict versions of the properties we can assure finite time convergence. Let us see what this means in terms of the protocols that can be modelled by such algebraic structures.

Considering the first case (strictly monotone ordered semigroup) it is well known that monotony is hard to maintain in policy rich protocols like BGP. This is mainly due to the existence of invalid paths [Sob05][Gri08][SG10]. Consider for example the set $S = \{p2p, c2p, p2p\}$ where $p2p$ is the value for links between peer nodes and $c2p$ and $p2c$ the values for customer to provider and provider to customer links. Also consider the

common policies in BGP that prevent paths learned from providers to be advertised to other providers. If one considers the usual preference of customer routes to provider routes we lose monotony since:

$$p2c \prec c2p \text{ but } c2p \otimes c2p = c2p \prec p2c \otimes c2p = \overline{0}$$

Or, in words, customer paths $p2c$ are preferred to provider paths $c2p$ but a provider path $c2p$ extended to another provider ($p2c \otimes c2p$) is invalid while extending a provider path to a customer ($c2p \otimes c2p$) is valid and it is still a provider path which is more preferred than the invalid path.

Although monotony is hard to maintain for policy protocols we have seen that convergence to a local optimum solution is still possible if the *increasing* property is verified. But in order to have an *increasing* $\otimes$ operation, the path weight has to strictly increase with any added link. This means that two paths where the same policy (link label $l \in S$) is applied to all links only have the same preference in $\preceq$ if they have the same number of hops. This is a limitation in the use of the possible path diversity in a network.

It would be interesting to have a $\otimes$ operation that can be merely *non decreasing* as defined in definition 4.8 instead of strictly *increasing*. This would allow paths with a different number of hops, but with the same policy on all links, to have equal preference and therefore increase the amount of paths that can be used simultaneously for forwarding.

The problem is that such an algebra does not guarantee convergence because $\otimes$ is not strictly *increasing*. With such an $\otimes$ operation we can have cycles in a network in which the path weight maintains preference with the addition of consecutive links around the cycle. Since the algebra by itself does not assure convergence, some extra restrictions on the network have to be set. Basically, we have to assure that problematic cycles are non existent in the network.

Such cycles can be formalized as a generalization of the *non negative* cycle condition. The *dispute wheel* in [GSW02] can be seen as a first formal description of these cycles. A more algebraic description is the generalization of the *non negative* cycle condition found in [Sob05] called a *free* cycle. The relationship between a *free* cycle and a *dispute wheel* of SPP is made in [CGG06].

We re-define the notion of *free* cycle in [GSW02] in a pure algebraic setting for an ordered semigroup in the following definition:

Definition 4.11. *Free cycle*

Consider a cycle $P = v_1, v_2 \cdots, v_{k-1}, v_k = v_1$. Consider d a destination, A_i the set of all paths from vertex v_i to d, and $\alpha_{ij} \in S - \overline{0}$ the weight of the j path starting at vertex v_i to destination d without going through v_{i+1}. Therefore, $\alpha_{ij} \in A_i$. Consider the sets of all paths to d from every vertex of the cycle, $A_1 \cup A_2 \cup \cdots \cup A_{(k-1)} \cup (A_k = A_1)$. The cycle is free if there is at least a path j starting at a node i $(1 \leq i \leq k-1)$ such that $\alpha_{ij} \prec w(v_i, v_{i+1} \otimes \alpha_{(i+1)k}), \forall_{k \in A_{i+1}}$. This has to be valid for all reachable destinations, d.

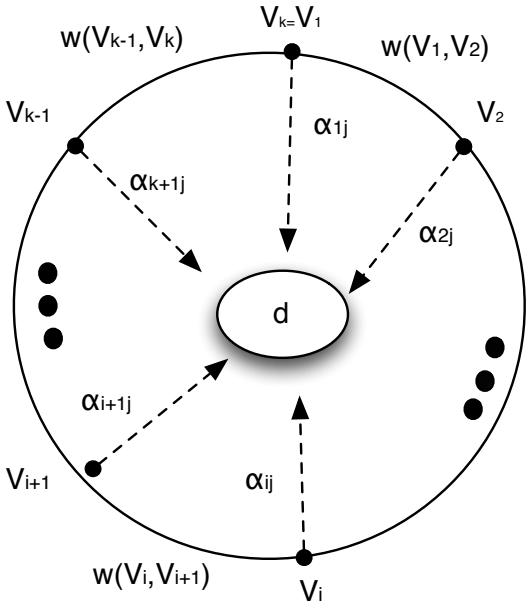

Figure 4.2: Free cycle illustration

Figure 4.2 illustrates the concept, for the cycle to be *free* at least one of the dotted line paths α_{ij} to d must exist and be more preferred at node v_i than the path in solid lines to $v_{i+1} \otimes$ added to the next $\alpha_{i+1,j}$ path, or other.

This means that it will always exist at least one path leaving the cycle that is more preferred than the ones across the cycle. Note that all cycles in strictly monotone, or at least strictly *increasing*, algebras are free as long as the network graph is labelled in a set S with lower bound $\overline{1}$ (*non negative* cycle condition).

In a network where all cycles are *free*, convergence in finite time is assured if the

algebraic model of the protocol is a:

- Monotone non-strict ordered semigroup. In this case the protocol converges to a set of best paths in finite time. This is because the problematic cycles where the preference would be maintained are *non free* and therefore do not exist in such a network.

- Non monotone ordered semigroups with *non decreasing* $\otimes$ operations. Here we have convergence to the minimal local optimum weight in finite time since best paths never traverse cycles. This follows from the fact that without *non free* cycles in the network the algebra is in fact *increasing* in cycles that do exist and this means that there will always be a shortest path out of a cycle.

In conclusion, restricting the network cycles provides more flexibility in the algebraic model properties and this brings the possibility to increase the number of paths that are considered of equal preference, which is precious when multipath routing is the goal. A final proof is needed: does a *non decreasing* algebra in a network with all cycles *free* converge when using a minimal set of paths instead of a single best path?

Multipath without strictness

The proof in [Gur09] for convergence in a multipath algebra demands a strict path value inflation with the addition of links. In a *non decreasing* algebra there is no guarantee that $\forall_{a,b \in S}\ a \prec a \otimes b$. However, it is possible to assure path value inflation in a network where all cycles are free. To prove this we consider the two possible types of path: acyclic and cyclic.

For simple acyclic paths path inflation is not necessary since a fixed point in the routing solution is always achieved in finite time (acyclic paths have a finite amount of edges).

For cyclic paths let's consider L the set of links that form a cycle c. If all cycles are *free* then for each cycle $\exists_{l_1,l_2 \in L}$ such that $l_1 \prec l_2 \otimes l_1$. This means that preference inflation will eventually occur around the cycle, thus assuring the path inflation condition needed for convergence in the proof of [Gur09].

An alternative proof can be taken from [Cha06]. [Cha06] studies multipath routing using a concept that relates the SPP framework with algebraic routing, called policy relation (introduced in [CGG06]). The author finds a similar conclusion to the one in [Gur09]: a multiple path routing solution is possible if the policy relation is anti-reflexive (an anti-reflexive policy relation can be mapped into an *increasing* algebra). What is interesting is that in [CGG06] the proof is based on the absence of *dispute wheels*, and these can be mapped on the *non free* cycles described above.

This absence of *non free* cycles can be achieved by the algebras properties (if it is *increasing*, *non free* cycles never exist) or by a combination of an *non decreasing* algebra and network graph constraints.

In conclusion, convergence to a routing solution in finite time can be assured for a protocol modelled by an *non decreasing* ordered semigroup provided that the network is constrained such that all cycles are *free*.

Up until now we have been discussing convergence in terms of converging to a routing solution in the sense that an algorithm obtains, in finite time, the best paths according to $\preceq$. After the routing solution is obtained another relevant issue is the forwarding of packets.

4.4.3 Destination based hop by hop convergence conditions

In this work the option is to maintain simple destination based hop by hop forwarding, and so it is important to see in which conditions we have loop free forwarding. Forwarding loops are caused by (independent) forwarding decisions that are not consistent. In a strictly monotone algebra, this is not a problem since sub paths of an optimal path are always optimal paths and therefore the decisions of the next hops will be consistent. If one drops monotony then a forwarding loop can occur. In this case the solution of equation (4.1) is a local solution depending on the preferred paths of neighbours. Here we have to distinguish equation (4.1), in which $\otimes$ is applied to the left, from the right local equation (equation 4.2) where $\otimes$ is applied to the right.

Both equations have the same solution in a monotonically ordered semigroup, but in the absence of monotony, solutions can be different. It is well known that right local

solutions can cause forwarding loops in destination based hop by hop forwarding [SG10]. The reason is that in equation (4.2) path weights are being calculated from source to destination, and since the best path for the next hop can be different, a forwarding loop can occur. In equation (4.1) the calculation is done from destination to source and even in the absence of monotony, forwarding loops do not occur since path weights are always dependent on the weights of the sub-paths from the next hops to the destination.

So, loop free is assured using destination based hop by hop forwarding: for monotonically ordered semigroups for either the right or left solutions, or for non monotone semigroups for the left local routing solution. In this latter case the solution exists if the operation $\otimes$ is *increasing*, or if it is *non decreasing* and the network has all its cycles *free*.

We derived the following method of checking for *non free* cycles from a method presented in [SG10] that we recast for the pure algebraic ordered semigroup model:

Definition 4.12. For each value $s \in S \setminus \overline{0}$ consider a set $L_s \subset S, L_s = \{l \in S \mid \text{such that } s = l \otimes s\}$. This means that a set L_s contains the link values that maintain the path preference with the addition of links. A cycle is *free* in a *non decreasing* algebra if *for each one* of the L_s, the cycle has at least one of its links not belonging to that set. Or inversely, a cycle is *non free* if *all* its links belong to one of the sets L_s (at least one).

4.4.4 Main results

In summary we have seen that known algorithms can convergence to a multipath routing solution in the following conditions:

1. The ordered semigroup is strictly monotone providing a global optimum solution without infinite paths (finite time convergence).

2. The ordered semigroup is just monotone providing a global optimum solution but possible infinite paths (infinite time convergence).

3. The ordered semigroup is not monotone but the $\otimes$ operation is *increasing* providing a local optimum solution without infinite paths (finite time) using the left local approach (applying $\otimes$ from destination to source).

4. The ordered semigroup is not monotone and the $\otimes$ operation is just *non decreasing* but the network is constrained having all cycles *free*. Provides a local optimum solution in finite time using the left local approach (applying $\otimes$ from destination to source).

Having a routing protocol modelled by a strictly monotonic ordered semigroup is ideal to assure convergence to a global optimum but has problems for policy routing and multipath. In a strictly monotonic ordered semigroup having a set S such that $\forall_{a \in S} \overline{1} \prec a$ implies that $\otimes$ is strictly increasing since:

If $\forall_{a,b \in S}\ \overline{1} \prec b$ then with monotony $a \otimes \overline{1} \prec a \otimes b$ which implies $a \prec a \otimes b$.

Moreover, there cannot exist $a, b, c \in S$ such that $c \otimes a \simeq c \otimes b$ or putting it in another way $\forall_{a,b,c \in S}\ \ c \otimes a \neq c \otimes b$. This means that the addition of a link cannot make two paths equally preferred, and also that the protocol cannot have dominant link values (i.e. links that define the entire weight of the path). So, two paths only have the same value if they are exactly equal in the labels of the hops and in that case if they have the same number of hops.

A second option would be to have a simple monotonic protocol. In this case convergence in finite time is not assured and forwarding loops are also possible. To avoid both situations the network cannot have *non free* cycles. We actually explore this possibility in chapter 8 and found that if a link is allowed to dominate the entire path weight (the reason why strict monotonicity was not possible in the first place) we would reach a situation where every cycle in the network would be *non free*. This means that such an algebra only converges in acyclic networks and such networks do not have multiple paths to the same destination.

Trying to have multipath maintaining monotonicity and a global optimum solution is weak: we either maintain strict monotony and have a very small number of equivalent paths; or we drop the strictness and easily end up with all cycles being *non free* which in practice means no multipath routing at all.

What happens if we drop monotonicity and accept local optimum solutions? In this case we have assured convergence for *increasing* $\otimes$ operations. This is still not the best choice. In this case hop count has to much influence in the paths weights, and this is not

a good thing in policy based routing. For routing in inter-domain like scenarios multipath routing is of interest since it allows a better approach to many of the issues discussed in chapter 3. In that context achieving equal preference for the maximum possible number of paths is of interest. The proposed system described in chapter 5 was therefore designed having in mind a protocol model with a *non decreasing* $\otimes$ operation meaning that the network is constrained not to contain *non free* cycles.

Chapter 5

Dynamic Topological Information Architecture (DTIA)

5.1 Introduction

From the analysis of the current inter-domain routing architecture of chapter 3 we isolated some characteristics that in our vision a routing architecture for the inter-domain scenario should have. We believe that routing should be kept as stable as possible and that other problems like TE should be solved on top of it. This means that a multipath routing architecture is desirable. It has several advantages and the obvious one is to make better use of the underlying network diversity. Other advantages come from the possibility of shifting traffic between paths without messing with route attributes or with the calculations and ranking of paths. This can be useful for performing TE in a routing independent way or to respond to failures in a seamless way.

In the inter-domain scenario each node in the network is managed in an independent way and without information about the configurations of other nodes. If a protocol allows the nodes to apply any policies they want without some limitations, convergence to a routing solution in finite time can not be guaranteed. The algebraic model would correspond in this situation to having each node calculating path weights according to different $\otimes$ operations and possibly using different preference orders $\preceq$ to order the weights. This is a problem that does not exist in intra-domain shortest path protocols such as OSPF or

IS-IS. If we look at them through a policy algebraic perspective all nodes apply the same $\otimes$ operation and $\preceq$ preference order preferring shortest paths.

In order to assure correctness some restrictions have to be set on the allowed polices both in terms of the identification of the policies that will be used, and in the fact that all nodes follow the same routing model. We will not use any signalling for this purpose and each node decides independently of the others. The aim is to have enough expressiveness in the allowed policies to model as many situations as possible for the current needs of a network.

DTIA follows this approach and assumes a certain abstraction for the network. For instance, a node is one AS and two nodes can only be connected by one link. This means that load balance mechanisms between ASes connected by more than one link cannot be performed at DTIA level. A set of policies is defined based on the types of links, setting forbiddingness of composition for certain sequences of links, and rules ranking the types of paths. An algebraic model expresses these three aspects for a set of concerns identified as important for the Internet network. It forms a closed system but the addition of new features is possible by redefining the set of policies, and the results of the operations for path composition and ranking.

DTIA is based on a three levels layered architecture: reachability, routing and upper layers. Paths are calculated at reachability level where they are assigned a weight according to the policies applied to their links. At a routing level an order of preference is used to rank the existent paths and choose the ones to use always assuming a destination based hop by hop forwarding. These operations assure a multipath stable routing solution on top of which we can perform other tasks to solve problems like TE. This separation reduces complexity and allows the modelling of the protocol so that we can prove its correct operation. The chosen way to calculate routes naturally deals with multi homing and provides a base for better failure handling. Scalability is dealt in two different ways: (a) routing per ASes instead of prefixes, providing a one time degree of reduction in the number of identifiers; (b) By creating an algebraic model that is amenable to the creation of regions to partition the network.

This chapter describes DTIA starting with the main assumptions and options behind

the design followed by a description in an algorithmically view of the protocol operation. We then model the protocol using an ordered semigroup algebra and prove its correctness. Finally we present a TE layer that operates on top of DTIA and present the results.

5.2 Design options

Instead of proposing disruptive business models, which are unlikely to be quickly adopted by the active players, our purpose is to define a system that changes the business model as little as possible but that will be able to evolve in the future. With that in mind we started defining the protocol taking into consideration the following design options.

Routing is based on AS connections and not on prefixes.

Working at prefix level enlarged the size of the routing tables and it is consensual that this growth must be contained [RFC07]. One way to reduce the growth rate is to rely on prefix aggregation. However, aspects like multihoming, load balancing and address fragmentation in the Internet have hindered prefix aggregation and as was discussed in section 3.2 of chapter 3 this leads to routing table growth and an increasing number of exchanged messages.

One of the discussed ways to reduce the growth is to route packets using AS labels. This decision is controversial with some opinions against it and others in favour since it has the advantage of separating routing from addressing but it requires the use of a mapping service between addresses/identifiers and ASes labels. The existence of mapping services starts to be considered each time more in the literature [SCE+05, D.07] and we decided to obtain the one time factor reduction in the routing table size via the use of AS numbers as routing locators.

All nodes use a set of common policy rules

Inter domain routing is based on routing policies. Paths are exported and chosen based on attributes that are set according to the policy objectives of the ASes. In BGP there is absolute freedom in attribute manipulation and therefore in the policy used. This flexibility and the lack of coordination between ASes impairs any protocol correctness guarantees.

Our system defines a set of policies that make up a clear policy part of the protocol.

These polices define how a path's weight (based on the commercial characteristics of its links) is calculated and how valid path weights are relatively positioned in a preference ranking. Nodes independently apply this set of polices. However, they all use the same order of preference and apply the same set of policies to links and paths.

Despite of all the flexibility of the BGP protocol, the commercial nature of the Internet has led to a disseminated use of routing policies based on the business relationships between ASes [CR05] forming a relatively small set. These *common policies* have an important part in the routing stability of the Internet, since their use has been proved to lead to a stable set of paths [GR01][HK07]. DTIA's policy part covers these business related *common policies* plus two extra ones that add flexibility to model some other types of relationships. The result is a stable set of paths with no forwarding loops. This can be proved by using an algebraic model of the rules and ranking system. As stated before new rules and ranking systems might be defined as long as they maintain the necessary properties that assure convergence to a stable set of paths. However, any change is considered a major change in DTIA because it needs changes on all nodes of the network. The algorithm is the same but the set of of possible link characteristics and path weights changes as well as the definition of the operations. As it stands in this thesis, DTIA does not address some functionality seen in BGP, like inbound traffic engineering. This is currently obtained by complex attribute manipulations like path prepending [CL05] or the use of communities [DB08] and it was decided to leave it outside of the routing algebraic model. The problem is not so serious because It can be performed above it.

Routers get a static map of the network and co-operate to learn about failures

Most of the routing events on the current Internet come from route announcements, and only a small part is due to transient topological failures [LGW+07]. In terms of topology, the structure of the network is somewhat stable [OZZ07], but new ASes and links appear every day.

In DTIA more importance is given to the handling of the dynamic part of the network that is caused by the seldom failures (and is a major problem in the current architecture) and not so much to the discovery of the graph of the network. Two options could be followed: the network map could be discovered via a traditional mechanism of route

advertisements; or the map can be distributed to the nodes. A third option would be to exchange link state packets but it is problematic in terms of scalability. In the option of route advertisements an AS would not know the entire map of the network but only the calculated paths that match the policy. Knowing the entire map has many advantages and one of them is that it can provide important information for TE purposes. For example in the proposed TE layer for DTIA an AS can use the map to infer the paths that other ASes have to itself. Therefore, we decided to adopt the second option. The distribution of the map could be accomplished with the exchange of link state packets, but since failures are going to have a specific treatment and we aim at an architecture with a small amount of churn a static map distributed via flooding was the selected choice.

We consider that a central entity (possibly replicated) delivers a static AS level map of the network (or a region, see the following design choice) to routers. Each time that is convenient, a new version of the map is sent. The RPSL (Routing Policy Specification Language) database of RIPE [RIP] is an example of a database that could serve to obtain the map. This means that new ASes and links only start to be used some minutes (or hours) after they become available.

The protocol assumes a static reality and builds a dynamic reality due to failures. A similar approach was followed with different purposes by [YCB07]. The dissemination of failure information should only disturb the relevant routers with precise rules about its scope.

Maps and co-operations are limited to regions

Most of the concerns in inter-domain routing are local to the ascending (and descending) paths. Real global events in BGP are again related to the withdraw procedure of prefixes. Depending on their placement in the hierarchy and what aggregations exist, events in BGP are confined to regions.

The overall topology at inter-domain level can have collections of ASes that according to their connections can provide some confinement for control traffic. DTIA uses this characteristic and defines the concept of a region as a scaling mechanism. Regions are not fundamental for DTIA, but they might be used for two reasons: scalability in terms of path calculation complexity and static map size; and system deployment (in this case it

is just transitory). Until we discuss regions we assume a region is just the network.

5.3 Architecture - general overview

DTIA has a three-layer approach. The first layer is concerned with reachability and the second with routing. Traffic engineering issues (e.g., controlling incoming traffic) can be performed on top of the routing level without impact on the available routes.

At the control plane the reachability and routing levels use the static network AS level map to produce the routes between ASes according to the policy. At the forwarding plane AS numbers are used to forward packets.

5.3.1 Control plane

Static AS Level Map The static network map is the base for the reachability and routing levels. It must be maintained by an entity with (possibly replicated) servers. The case of RIPE's RPSL databases [RIP] was already referred. Several studies that tried to infer AS level maps of the Internet [DKF+07, Gao00, OPW+10] have shown that information is incomplete and sometimes inaccurate. This is due to the lack of precise information on the type of link between the ASes, provided by each AS. This information is considered confidential but in practice all stake holders end up knowing the relationship in terms of routing policy (and not the business agreement that supports it of course). By advertising routes in BGP ASes are giving already the information DTIA needs and it is just a question of having an entity that collects it and announces it. The only issue is to provide proper incentive for ASes to maintain the databases. Once the map exists, there must be a way to distribute it. Flooding can be used. It is not that inefficient due to the small world characteristics of the Internet. During bootstraping the map can be requested by ASes via their provider links. New versions of the map are distributed to tier-1 ASes (possibly with a direct link) and each AS disseminates it immediately downwards and through peers. Given the small world effect, naive algorithms such as a version check prior to its transmission can be performed. Transient forwarding loops might occur during the distribution of new map versions in the same way they can occur during the distribution of failure information (that causes different map versions to be present in the network).

These loops are however confined (see the failure handling description in Section 5.5).

Reachability Level - Calculating path weights After the reception of a map, the AS starts the process of calculating the weight of the paths.

The first step, performed at the reachability level [ABP09], is to apply the policy rules to calculate the paths weights. A path can be invalid (weight $\overline{0}$) if it contains a succession of links that due to their characteristics violate the policy. As stated above, DTIA policy is based on the so-called *common policies* extended with two more policies that support some other possible network scenarios. The *common policies* comprise the provider-customer and peer-to-peer relationships that are enough to deal with 99% of the relations used today in the Internet [Gao00, SCE+05]. The extra two are used for sibling-like relationships and backup specific relations (as suggested in RFC 1998). The reachability level produces something similar to the result of a path vector protocol, routing on AS numbers, that exports not only the best path to a given destination but all paths that comply with the policies. The difference is that in DTIA the process is local to each AS and no routing messages are exchanged. Also, the identification of all paths to a destination creates the multipath routing characteristic.

Routing Level - Path Ranking In DTIA we propose a multi-path business routing protocol, which assigns an order of preference amongst the valid path weights. This preference is based on the business properties of the path, for example a path can be a customer or a provider path and this qualifier is its weight. The path weight is calculated according to the business relationship of the links that form it at the reachability level. The preference order then determines which are preferred and used for forwarding.

We model DTIA in an algebraical form using an ordered semigroup as defined in definition 4.5 of section 4.4 in chapter 4. We use the model to prove that DTIA will work correctly. DTIA does not exchange routing messages in the control plane so no routing loops in the traditional sense occur. However, in order to be correct we have to prove that the route calculation process converges to a set of local optimal paths in finite time. DTIA uses simple destination based hop by hop forwarding and therefore we must also prove that no forwarding loops occur. The final result at the routing level is a forwarding table

based on AS numbers. This table can contain more than one path to each destination and separates them according to their preference.

Intra AS route distribution As stated in the introduction of this chapter we have only one entity per AS, abstracting the AS internal structure. We also assumed a single connection between adjacent ASes even if they have more than one physical link. The reality inside of the AS (intradomain) is not the focus of this thesis but some guidelines are the following:

ASes have DTIA (border) routers and internal routers; an internal routing protocol is running using IP addresses; each router knows if an IP address belongs to the AS, or not; DTIA routers exchange the DTIA map between them together with a physical information about their physical links. When a router has a packet to an external destination it sends it to the nearest DTIA router. The mapping to AS number is performed by this router and the packet is sent to the precise DTIA router in this AS or to another AS if this DTIA router is the one to be used. Note that this relationship between the intra and inter realities can be improved by extending DTIA in several directions.

5.3.2 Forwarding plane

Forwarding is performed by choosing one of the available paths with the most preferred weight. The possibility to use multiple paths of the same business category without causing forwarding loops provides a good base for Traffic Engineering. A proposal for traffic engineering using DTIA that has concerns about congestion and inbound traffic control is described in chapter 6.

Since the forwarding is based on the AS numbers, all routers (ASes) along the path need to perform the mapping between IP addresses and AS numbers. This is needed to maintain the IP packet unaltered. A possible optimization is to prepend a header in the packet containing the destination AS number eliminating the need for mappings along the path. This header would be added by the border router of the originating AS and removed by the border router of the destination AS in a similar way to what happens in the Locator /ID Separation Protocol (LISP) [D.07].

5.3.3 Mapping

Routing by AS numbers provides a reduction of the routing tables because of the lower order of magnitude of ASes compared to IP addresses and because hosts do not need to advertise more than once to feature multihoming or traffic engineering. The mapping service does the aggregation "magic" of converting from a larger set (IP addresses) to a smaller set, and depending on the features it contains (see end of this section) it might be quite complex. Mapping services start to appear in different systems: they can be intrinsic to the routing algorithm, as in the compact routing proposals [KcFB07], or be independent with different degrees of autonomy.

Two systems, HLP and LISP, are similar to ours in this aspect and consider the existence of mapping services. HLP [SCE+05] provides little detail on how their service works (except that it is a mapping table). LISP [D.07] defines a set of messages and procedures (subsets, priorities, etc.) to implement the service. LISP describes four variants for the role of the routers in the service and the highest one (LISP 3) assumes that an external service performs the mapping. One example is LISPDHT [MI08] which is a hybrid push and pull model based on distributed hash tables (DHT). DHT provides interesting features such as self-configuration, self-maintenance, scalability, and robustness.

A complete proposal for a mapping service is outside of the scope of this thesis: A possibility that is highly distributed is to consider border routers as Distributed Hash Tables (DHT) servers (or acting on behalf of them). When a packet arrives to a border router for a different AS, a hash value (calculated from its IP address or prefix) determines the AS number responsible for the resolution. A request is then sent to that AS and the reply sent by one of its routers transport the (set of) AS numbers to be used for forwarding the packet. This solution is based on a pull mode. The cost of operating a mapping service is still quite unknown and [IB07] evaluates the cost for a pull model. Based on it, we believe that the dynamics and query traffic are comparable but smaller than the DNS case, which provides a good indication in scalability terms.

LISP had very constrained requirements in order to change as little as possible the router software and hardware. Therefore the protocol is very similar in functionality to the IP protocol. It is relevant to state that if the changes are not a concern useful features

can be placed on the mapping service such as: (a) an anycast address/identifier could be mapped to different ASes' routers; (b) a TimeToLive parameter to control the lifetime in the cache is important to support dynamism or node/network slow mobility; (c) multiple reply values can provide support for node-multihoming (other forms of multihoming are supported already by DTIA); etc.

5.4 DTIA's algebraic model

The region static map $G(V, E)$ is modelled as a directed graph with V vertices that model ASes and E edges that model links between ASes. The routing protocol is then modelled by an ordered semigroup.

$$(S, \preceq, \otimes)$$

The edges are labelled according to the commercial relationships between the ASes with elements $l \in S$. We consider four types of inter-AS relationships:

Provider-Customer. One AS (the provider) accepts all traffic from the other AS (the client). The edge has a label ($p2c$) in the provider-customer direction and another ($c2p$) in the customer-provider direction.

Peer-to-peer. ASes provide connectivity for their direct or indirect customers. No transit traffic from the peer is allowed. The edge is labelled ($p2p$) in both directions.

Peer-to-peer allowing backup. The same as before but allows transit traffic if no other path exists. The edge is labelled ($p2pbk$) in both directions.

Peer-to-peer allowing transit traffic. Transit traffic is allowed in any situation (we use this to model sibling relationships). The edge is labelled ($p2patt$) in both directions.

The calculation of path weights is modelled by the binary operation $\otimes : S \times S \longrightarrow S$ that maps the labels of the edges that form a path to a path weight $q \in S$. Certain combinations of edges are considered invalid and therefore those paths have weight $\overline{0}$ which is the least preferred weight of all so that they are never used. The set S is composed by the following elements:

- $c2p$ label of an edge in the customer to provider direction or for a path that starts with a neighbour AS that is a provider.

- $p2c$ label of an edge in the provider to customer direction or for a path that starts with a neighbour AS that is a customer.

- $p2p$ label of an edge between peers or for a path that starts with a peer AS.

- $p2patt$ label of an edge between peers allowing transit traffic or for a path that starts with a peer that allows transit traffic.

- $p2pbk$ label of an edge between peers that can be used as part of a backup paths.

- bkp for a backup path, a path that can only be used if no other paths are available for the destination.

- $\overline{0}$ for an non existent or invalid path.

- $\overline{1}$ for the trivial, single node, path.

After the path calculation, several valid paths might be available for each destination, each one with a certain weight value (element of S). The order of preference between the weights is modelled by the order $\preceq$ of the ordered semigroup modelling the protocol. Paths with the same weight or paths that are considered equivalent according to the $\preceq$ order can be used simultaneously. Before going further into the protocol description it is useful to discuss some general aspects. As we mentioned before, routing loops, as they are traditionally defined (an infinite exchange of routing messages without achieving a stable routing state) do not occur in DTIA since no messages are exchanged. But a problem still exists if a stable routing state is not achieved in finite time. This can be guaranteed if the path composition operation and the preference order have the right characteristics. The following section provides an algebraic model of DTIA and analyses this aspect.

Another kind of loop is the forwarding loop. A forwarding loop happens when a data packet fails to progress to the destination by traversing over and over the same links. When destination based hop by hop forwarding is used the properties of the path composition operation and the preference order are also important. It is so because we need something to assure that the forwarding decisions of ASes along a chosen path do not cause *forwarding loops*.

To see the *forwarding loop* problem consider the very simple topology of Fig. 5.1. AS C has two valid paths to destination AS G: C-D-G and C-E-G. AS D also has two valid paths, D-G and D-C-E-G.

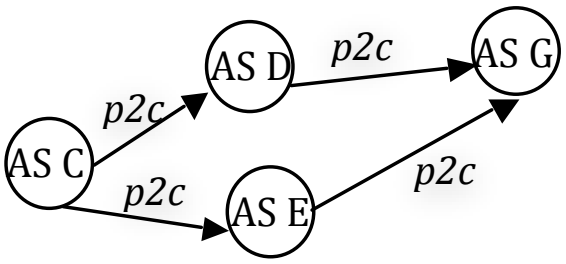

Figure 5.1: Simple topology

Forwarding loops can occur if, for instance, AS C chooses the path C-D-G and D chooses the path D-C-E-G to reach G.

The use of a preference order by all nodes will be key in preventing this. The preference order is based on three rules – one is already well-known in the current Internet (as part of the *common policies* [CR05]) and the other two are related to our extensions:

1. Customer routes, *c2p*, are preferred over peer, *p2p*, or provider, *c2p*, routes.

2. Primary paths are always preferred over backup *bkp* paths.

3. Amongst primary paths, *p2c* paths are preferred followed by *p2p* paths and *c2p*. *p2patt* paths have the worst preference.

The algorithm to calculate the weights takes into account the links that form the path. Therefore, the weight reflects the characteristics of a path in terms of business policy. We will prove that if each AS along the path uses paths belonging to the highest preference level available the protocol operates correctly.

5.4.1 Protocol correctness

Informally a protocol is correct if in a stable network (with no changes occurring) it determines a stable set of paths in finite time at the control plane and produces no forwarding loops at the forwarding plane.

Table 5.1: DTIA's $\otimes$ operation

$\otimes$	$\overline{1}$	$p2patt$	$p2c$	$p2p$	$p2pbk$	$c2p$	(bkp, x)
$p2patt$	$p2patt$	$p2patt$	$p2patt$	$p2patt$	$(bkp, 1)$	$p2patt$	$(bkp, x+1)$
$p2c$	$p2c$	$p2pc$	$p2c$	$\overline{0}$	$(bkp, 1)$	$\overline{0}$	$(bkp, x+1)$
$p2p$	$p2p$	$p2p$	$p2p$	$\overline{0}$	$(bkp, 1)$	$\overline{0}$	$\overline{0}$
$c2p$	$c2p$	$c2p$	$c2p$	**c2p**	$c2p$	$c2p$	$(bkp, x+1)$
$p2pbk$	$p2pbk$	$(bkp, 1)$	$p2pbk$	$\overline{0}$	$(bkp, 1)$	$(bkp, 1)$	$(bkp, x+1)$

In chapter 4 the necessary conditions for protocol convergence are discussed. In DTIA's case we are interested in proving that a solution for the routing problem is possible in finite time and also we need to check if no forwarding loops occur.

DTIA is modelled using an ordered semigroup of the form $(S, \preceq \otimes,)$. S is a set that contains the possible link labels and path weights, $\prec$ defines the *preference relation* over the elements of S (modelling the ranking process of the routing level of DTIA). The $\otimes$ binary operation models the path composition operation (the reachability part of DTIA) and it is applied between two elements of S, a path weight and a link label resulting in a new path weight. It expresses in algebraic form the DTIA algorithm that assigns a given weight to a path, based on the labels of its links.

We have the set $S = \{\overline{1}, p2patt, p2c, p2p, c2p, p2pbk, bkp, \overline{0}\}$, that contains the domain of possible link labels and path qualifiers. Their meanings are the ones listed in the beginning of section 5.4.

The operation $\otimes$ defined in table 5.1 models the path composition operation. The composition consists on appending a link of type $l \in S$ to a path with a certain weight $p \in S$ resulting in a new path weight. The path starts to be formed at the destination with links being appended in the source direction. The type of the link is the type it has in the source-destination direction. For instance, consider the grey cell in Table 5.1 with bold font. The meaning is that a path with a weight $p2p$ can be extended in the direction of the source by a link with label $c2p$. The weight of the path becomes $c2p$. This represents a link coming from the source in the ascending direction (to a provider) followed by a path that has a $p2p$ weight from that point to the destination.

Table 5.2 shows the preference order for the paths weights. The higher the more preferred.

Table 5.2: DTIA's order of preference

$p2c$
$p2p = p2pbk$
$c2p$
$p2patt$
$(bkp, 1)$
...
(bkp, n)

A peer-to-peer link that can also be used as a backup is labelled $p2pbk$. These links can either be used as a regular peer-to-peer link or as a part of a backup path. Such flexibility has consequences for the preference rules. In the former case the resulting weight for the path to which this link is appended is $p2pbk$ (and should have the same preference as a $p2p$ path). When it is used as part of a backup path the result is a weight (bkp, y) with y getting strictly increasing natural numbers as new links are appended. Two concrete examples from Table 5.1 for this type of links are:

Backup links used as normal peering: consider the example, $p2pbk \otimes p2c = p2pbk$. It means that a path from a customer is extended to a peer, this is a normal peering relationship and therefore the resulting weight is $p2pbk$.

Backup links used as backup: for backup paths the resulting weight is (bkp, y). The value of y increases every time a $p2pbk$ link is used for transit traffic. For instance, $p2pbk \oplus c2p = (bkp, 1)$ means that an AS can transit traffic between a peer and a provider in a backup situation. The path starts by having $y = 1$. For every new link in a backup path the integer is increased. For instance, $p2c \oplus (bkp, y) = (bkp, y + 1)$ means that extending a backup path to a provider is possible but decreases its preference.

This y value is a secondary routing metric that applies only to bkp paths. Generically, two different routing metrics can be modelled in an algebraic form by taking the lexicographic product of the two algebras (one for each metric) [BT10b] [GM08] [GG07]. In this case we have a lexicographic product between the ordered semi group $(S, \preceq, \otimes)$ and a secondary ordered semigroup $(\mathbb{N}, <, \otimes_n,)$ formed by the naturals set $\mathbb{N}$; an $\otimes_n$ operation that is the addition operation $+$ between the values $y \in \mathbb{N}$ for paths with a bkp value in S (it is not applied to other path weights), and the total order $<$ that is induced in the

naturals by the *min* operator. This secondary structure models a secondary path attribute only applied to *bkp* paths. The total algebra is given by the lexicographic product:

$$S \overrightarrow{\times} N = (S \times \mathbb{N}, \preceq_{S \overrightarrow{\times} N}, \otimes_{S \overrightarrow{\times} N})$$

where

$$(s_1, n_1) \preceq_{S \overrightarrow{\times} N} (s_2, n_2) \iff s_1 \prec s_2 \vee (s_1 \simeq s_2 \simeq bkp \wedge (n_1 \leq n_2))$$

and

$$(s_1, n_1) \otimes_{S \overrightarrow{\times} N} (s_2, n_2) = (s_1 \otimes s_2, n_1 + n_2)$$

This means that for *bkp* paths, and only for these, preference depends on the number of hops. To prove the protocol correctness one must prove that the protocol will converge to a set of stable routing paths in finite time and that it does not produce forwarding loops.

Proof. We have showed in chapter 4 the necessary conditions for protocol correctness. They were:

1. The ordered semigroup is strictly monotone providing a global optimum solution without infinite paths (finite time convergence).

2. The ordered semigroup is just monotone providing a global optimum solution but possible infinite paths (infinite time convergence).

3. The ordered semigroup is not monotone but the $\otimes$ operation is *increasing* providing a local optimum solution without infinite paths (finite time) using the left local approach (applying $\otimes$ from destination to source).

4. The ordered semigroup is not monotone and the $\otimes$ operation is just *non decreasing* but the network is constrained having all cycles *free*. Provides a local optimum solution in finite time using the left local approach (applying $\otimes$ from destination to source).

If we look at the DTIA model we can see that the ordered semigroup $(S, \preceq, \otimes)$ is not strictly monotone since for example $c2p \preceq p2c$ but $p2patt \otimes p2c = p2patt = p2patt \otimes p2c =$

p2patt. This means that convergence to a routing solution in finite time is only possible in networks without *non free* cycles. An alternative to assure convergence in finite time would be to have an *increasing* $\otimes$ operation (condition (3)) .

We can see in Table 5.1 that the $\otimes$ operation is merely *non decreasing*. In $\otimes$ a path with a certain weight extended by a link/label never results in a path with a more preferred weight. This is easily visible if we analyse each column: the result is similar or less preferred than the weight on the first row. $\otimes$ is simply *non decreasing* because some paths keep the same preference when extended. So, we can assure convergence to a local optimum routing solution in finite time, and that there are no forwarding loops if a left local solution is calculated and the network does not contain *non free* cycles (condition (4)).

However, and since the total model for DTIA is a lexicographic product

$$S \overrightarrow{\times} N = (S \times N, \preceq_{S \overrightarrow{\times} N}, \otimes_{S \overrightarrow{\times} N})$$

we also need to check if the lexicographic product is also under condition 4. In [GG07] the lexicographic product is studied, and theorems are derived that relate the algebraic properties of the individual algebras to the properties of their lexicographic product. According to [GG07] the lexicographic product $S \overrightarrow{\times} \mathbb{N}$ is *non decreasing* (ND) if :

$$ND(S \overrightarrow{\times} \mathbb{N}) \Longleftrightarrow I(S) \vee (ND(S) \wedge ND(\mathbb{N}))$$

where $I(S)$ means that S is *increasing* and $ND(S)$ means that it is merely *non decreasing*.

The ordered semigroup $N = (\mathbb{N}, <, +)$ is increasing since $\forall a, b \in \mathbb{N}$ we have $a < a + b$ which means that it is also *non decreasing*. Since S is only *non decreasing*, $S \overrightarrow{\times} N$ is *non decreasing*. Therefore the DTIA algebraic model falls under condition 4 which means that the protocol converges to a local optimum routing solution and produces no forwarding loops if a left local solution is calculated. However, in a *non decreasing* model the network must be constrained not to have *non free* cycles, so the final step to prove DTIA's correctness is to find the possible *non free* cycles. Definition 4.12 provides a way to check this by analysing the algebra for possible *non free* cycles. Basically, the idea is to

verify the link label values that maintain preference according to $\otimes$ around a cycle. Then, we must define for each element $s \in S \setminus \overline{0}$ the set $L_s \subset S, L_s = \{l \in S \mid \text{such that } s = l \otimes s\}$ of all elements of S that $\otimes$ added to s maintain preference.

For DTIA we have:

- $L_{\overline{1}} = \{\}$ because there is no element $l \in S$ such that $\overline{1} = l \otimes \overline{1}$.
- $L_{p2patt} = \{p2patt\}$ because only for $l = p2patt$ we have $p2patt = l \otimes p2patt$.
- $L_{p2c} = \{p2c\}$. Since only $p2c = p2c \otimes p2c$.
- $L_{p2p} = \{\}$ because there is no element $l \in S$ such that $p2p = l \otimes p2p$.
- $L_{p2pbk} = \{\}$. There is no element $l \in S$ such that $p2pbk = l \otimes p2pbk$.
- $L_{c2p} = \{c2p\}$. Since only $c2p = c2p \otimes c2p$.
- $L_{bkp} = \{bkp, p2patt, p2c, c2p, p2pbk\}$. Since for all elements of $s \in S$ except $p2p$, $bkp = l \otimes bkp$.

These sets are only concerned with the ordered semigroup $(S, \preceq, \otimes)$. The total algebra is the lexicographic product $S \overrightarrow{\times} N$ and according to the definition of $\otimes_{S \overrightarrow{\times} N}$ all sets remain equal for the total algebra except the L_{bkp} set that according to $\otimes_{S \overrightarrow{\times} N}$ becomes:

- $L_{bkp} = \{\}$. Since for all elements of $s \in S \overrightarrow{\times} \mathbb{N}$, $(bkp, n) \neq l \otimes_{S \overrightarrow{\times} N} (bkp, n)$.

A network has a *non free* cycle if it contains a cycle with all links belonging to one of the above sets. More specifically, a cycle is non-free in the following cases:

1. All its links have labels $p2patt$.
2. All its links have labels $p2c$.
3. All its links have labels $c2p$.

Our objective is the opposite: if the network does not contain cycles formed by any of these sequences of links, DTIA converges. Let's analyse each one of the cases.

Cycles in the 2 and 3 cases are formed only by $p2c$ or $c2p$. These are assumed not to exist in the Internet since they configure a cycle in the provider customer hierarchy. It

would mean that a provider would have a direct or indirect provider that is a client of one its clients. It does not make sense in the current Internet business model. However, the possible existence of such cycles can also be verified by checking the static map. Case 1 refers to a cycle of *p2patt* links. This is also a situation that is assumed not to exist. We do not foresee a scenario that could be modelled by allowing cycles with *p2patt*. However, if they have to exist we want them to be *free* and so we must apply a second metric to *p2patt* paths with a lexicographic product similar to what was done for the backup links (*bkp*) case.

This concludes the proof that DTIA converges to a local optimum left multipath routing solution in finite time and that it is forwarding loop free. The only condition is that network has to be free of cycles of types 1;2;3. □

It is now possible to assess the impact of changing the rules/policies of DTIA. It is possible to change the weight and link type set S, the $\otimes$ operation and the preference order $\preceq$. However, to maintain protocol correctness the resulting model must have the desired algebraic properties. The challenge resides on the usefulness of the definition of new link types to support new policies for routing.

Another interesting insight that the proof provides concerns the reason why a peer path allowing transit traffic (*p2patt*) had to be considered less preferred than customer, provider and peer paths. Apparently and according to the business model it makes no sense that a path via a peer allowing transit is less preferred than a provider path. But if we choose it to be of equal preference we end up with a protocol where cycles formed by links with labels $l \in \{c2p, p2patt\}$ are *non free* cycles. Let's see why: firstly if we consider $p2patt \preceq c2p$ then $p2patt \otimes c2p$ can no longer result in *p2patt* as defined in table 5.1 since this would break the *non decreasing* condition for $\otimes$ and therefore the protocol would not converge to a routing solution. The logical result for $p2patt \otimes c2p$ would have to be *c2p* for a protocol where $p2patt \preceq c2p$. This would assure that $\otimes$ is *non decreasing* but what are the consequences in terms of possible *non free* cycles? With these changes, the set L_{c2p} is now equal to $\{c2p, p2patt\}$ because for $l = p2patt$ we have $l \otimes c2p = c2p$. And more, *p2patt* is more preferred than *c2p*. This means that a simple cycle like the one in figure 5.2 woud be *non free*. The worst problem is the occurrence of forwarding loops: AS

B would find the paths $B - A$ and $B - C - A$ both with weight $c2p$ and therefore equal preference. A similar state would occur in C with paths $C - A$ and $C - B - A$ with the same preference $c2p$. So, a forwarding loop can occur in this case between B and C.

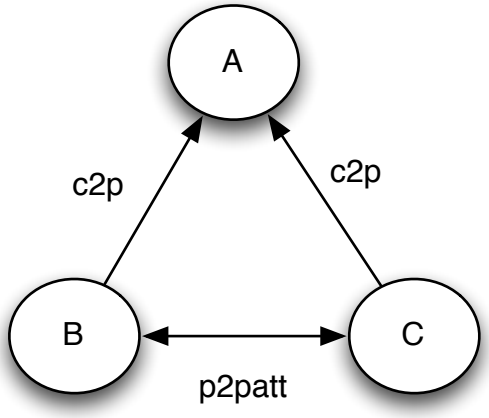

Figure 5.2: *Non free* cycle example

Reducing the preference of $p2patt$ and changing the result of $p2patt \otimes p2c$ to $p2patt$ solves the problem. Solving the problem without changing the preference would imply coordination between B and C so that only one of them prefers the path via the peer. In DTIA we want to maintain simple destination based hop by hop forwarding without any coordination and therefore chose to reduce the preference of the $p2patt$ paths.

5.4.2 Algorithm complexity

DTIA's proven correctness means that an algorithm that calculates the paths by applying $\otimes$ and then minimizes the preferences according to $\preceq$ will converge to a solution in finite time. We implemented the DTIA's algorithm using a solution that performs a depth-first exploration on the AS network map. The algorithm is optimized using some heuristics like stopping the exploration of paths resulting in a $\overline{0}$ weight and recognizing common segments of the paths. A brief description of the algorithm can be found in Appendix A. Depth-first algorithms have linear time complexities that increase with the number of vertices and edges. However, DTIA's algebraic properties reduce the problem's complexity in two different ways.

Consider that the AS map is modelled by a directed graph with a set V of vertices and

a set E of directed edges, $G(V, E)$. The time complexity for a depth first search algorithm is $O(|E| \cdot |V|)$. This time complexity is reduced in the DTIA algorithm.

A first reduction is due to the fact that not all combinations of edges that form paths have to be explored. For example, consider a vertex $N \in V$ that is reachable via two different paths P_1 and P_2 with the same weight $s \in S$. Consider that P_1 was already explored and E_N is the set of all edges that can be appended both to P_1 or P_2 after node N. Any path formed by appending a link belonging to E_N to the path P_2 will have the same weight to the corresponding one appended to P_1. This is because both have weight s when they arrive at N and the result of DTIA's $\otimes$ operation with the link is the same. Consider now $|S|$ to be the cardinality of S corresponding to the number of possible weights (elements of S) of the algebra. The worst case scenario in terms of path exploration is $|S|$ paths after vertex N. The maximum number of explored paths in the worst case is therefore $|S| \cdot |E|$ where $|E|$ is the number of edges in G, and it occurs if all vertices of G can be reached by $|S|$ paths all with different weights. The worst case for the path length is when a path is a Hamiltonian path reaching all $|V|$ vertices of the graph. In that case the total time complexity of the DTIA algorithm is:

$$O(|S| \cdot |E| \cdot |V|)$$

However, this worst-case scenario cannot occur in a DTIA labelled graph for two reasons: one reason is that if the worst-case scenario for path length occurs then the worst-case scenario for the number of paths does not. It is simple to verify - if all paths were Hamiltonian after $|S|$ paths had visited $|V|$ nodes all new paths would be equal to the existing ones and not explored; in fact there would only be $|S|$ paths. The other reason is described in the next paragraph.

The second complexity reduction characteristic of DTIA is that not all paths are valid due to the policies. This is visible by the $\overline{0}$ entries in table 5.1. This means that the worst-case $|S| \cdot |E|$ for the number of paths is not possible.

5.5 Failure management

The static map is no guarantee that the links are up. The dynamic part of the protocol is used to create awareness on link failures. There are two goals: assure that no forwarding loops occur during failures and that no packets are lost if at least a failure free path exists to the destination.

Only links fail (a failing AS means all its links failed). Assume a link fails. Routers disseminate a control packet that contains the identification of the failed link plus all links this AS knows that have failed (to consider the effect of multiple failures), and the version number of the map. When an AS receives a control packet only identifying failed links it already knows about, the packet is discarded. Control packets are acknowledged to ensure reliable delivery.

When a control packet arrives (or the failure of the link is detected in the case of the first router) the AS checks if it can still reach all reachable ASes without using the failed link. If at least one reachable AS becomes unreachable the control packet dissemination continues. If all ASes remain reachable the protocol evaluates the effects of the failure in the forwarding decisions. It can happen that after the failure the new paths to some destinations have a lower preference level than the original ones, and forwarding loops might exist (since we are outside the conditions in which the algebra's proprieties that assure correctness are valid). If this happens, a control packet must still be issued. The existence of changes is verified by running the DTIA's algorithms on the new map (without the failed links). Therefore, the processing time that we evaluate in chapter 7 is a constraint that must be considered.

Consider the example in Fig.5.3. AS C has a $p2pbk$ link with AS D, AS D is a customer of AS G and AS E is a customer of both AS C and AS G. If a failure occurs in link C-E all ASes remain reachable through valid paths from both C and E. However, before the failure the path weight from C to E was $p2c$ (the direct path). After the failure the weight is $(bkp, 1)$ (C-D-G-E path), AS C knows about the failure and starts to route packets to E through it. But AS D is unaware and will continue to choose the $p2pbk$ path (the D-C-E path) over the least preferred $c2p$ path (D-G-E). So, a forwarding loop occurs between AS D and AS C during the failure.

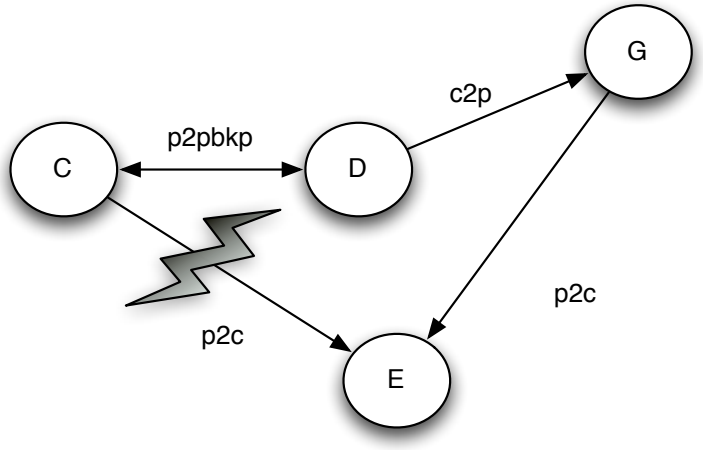

Figure 5.3: Failure example

AS D must then be notified about the failure because AS C changed its selected path to a path with a lower preference than the original one, and in the network map known to AS D (still including the failed link) this goes against the $\preceq$ order of DTIA's routing algebra. As soon as AS D is notified about the failure it will start to use a map with this information and DTIA's convergence properties are assured.

The failure notification protocol can be summarized by the following actions: the protocol checks if there are reachability changes, if it is the case, control packets are immediately disseminated; if all ASes are still reachable it then checks if the forwarding paths changed to a weight that is lower in the preference order. In this case, control packets are sent.

The dissemination is stopped in combinations of links that result in invalid paths according to $\otimes$. For example, an AS that detects a link failure to one of its providers does not send a control packet to another provider since in this case the failed link is already invalid (since $p2c \otimes c2p = \overline{0}$).

When the link comes up again a similar algorithm is used to disseminate a control packet with the link up information. I.e., dissemination continues if an AS that was unreachable because of this link becomes reachable, or if a more preferred path starts to be used with the correction of the failure. The dissemination uses reliable sessions.

The fact that only the relevant ASes receive control packets is a very powerful contention mechanism. The scope of the dissemination is directly related to the degree of

multihoming in the region. A high degree of multihoming makes the disseminating region smaller (there is less losses of reachability to some AS); and in terms of routing, more alternative paths with the same level of preference will exist (e.g. two $c2p$ paths or two $p2c$ paths) stopping the dissemination.

If the link of a single-connected AS fails, the dissemination always reaches the entire region (this AS becomes unreachable for everybody). Note that this is what happens today in BGP for a prefix that is single-homed and the link fails. We can approach this situation in DTIA in two steps:

1. wait an interval to see if the link comes back in service.

2. do one of the following:

   (a) send the routing event (it will go everywhere);

   (b) inform the entity that distributes the map that the AS no longer exists in case there were plans for a considerable time of non-activity.

Note that during step (1) packets directed to the failing AS are sent just to reach the failing AS provider and discover that the AS no longer exists. It is not desirable but this lack of delivery guarantee is consistent in the current Internet. Even today packets can reach a destination AS just to know that the prefix might not be valid at that moment but for some reason it is still advertised or not yet redrawn.

The update of map versions has also implications in terms of possible forwarding loops since different maps are available at each AS. Forwarding loops might occur and their nature is similar to the failure case: the new map might cause either a change in the reachable ASes or a change in the level of preference for certain paths. The reception of a new map will trigger a synchronization handshake with selected neighbors and the situation is solved similarly to the control packet's case. The implications in DTIA's routing correctness are the same as for the failure case and are explained in the next section.

5.5.1 DTIA's correctness in presence of failures

To prove the correctness of the routing protocol when failures occur we start by proving that every concerned AS is informed (both in terms of reachability and routing). Then, we prove that transient loops are contained and do not survive the dissemination of the control packets. Finally, the last theorem proves that if there is a path to the destination no packet is lost.

Theorem 5.1. *The control packet dissemination mechanism is guaranteed to inform every AS that experiences the following: a previously reachable AS becomes unreachable due to the failure.*

Proof. The AS detecting the event, x_1, informs every neighbour directly connected to it, x_2. The control packet wave continues hop by hop until it reaches ASes in hop n, AS x_n, for which all reachable ASes still remain reachable. Assume now that an AS at hop $n+1$ should receive a control packet and it does not. This can only happen if:

1. AS x_{n+1} loses reachability to some other AS (it should receive the control packet), and

2. All its x_n neighbours have alternative paths around the failure and reach all reachable ASes (and decided not to send it).

If all x_n neighbours have alternative paths to every reachable ASes the x_{n+1} AS will not lose reachability because it can use one of its neighbours. The first option represents a failure in one of the direct links between AS x_{n+1} and the x_n ASes but in this case AS x_{n+1} detects the failure and starts the dissemination of a control packet. Therefore, condition 1 and 2 cannot occur simultaneously. □

Theorem 5.2. *The control packet dissemination mechanism is guaranteed to inform every AS that has to change routing decisions.*

Proof. Let G be the region static graph and $DG(t)$ the region dynamic graph at time t (i.e., considering the failed links up to instant t). R_n is the set of all reachable ASes from a given AS n. $RD_n(t)$ is the set of current paths at time t being used to reach a destination AS D from AS n.

An AS (X_1) detecting a failure or a link up event checks if for all AS D $\in R_{X_1}$, $RD_{X_1}(t)$ has the same weight as $RD_{X_1}(t^-)$ (the path or paths used to destination D just before the event). If not, it sends a control packet to its neighbours. This control packet is forwarded hop by hop until a AS $n-1$, (X_{n-1}), for which $RD_{X_{n-1}}(t)$ has the same weight as $RD_{X_{n-1}}(t^-)$. At this point the dissemination is stopped and so X_n does not receive a control packet.

The path weights are calculated according to the operation $\otimes$ defined in Table 5.1. At X_n for all ASes D reachable through any X_{n-1} the weights of the paths at time t, $RD_{X_n}(t)$ result from using $\otimes$ to combine the label of the link $X_n - X_{n-1}$ with the weight of the paths to D in X_{n-1}, $RD_{X_{n-1}}(t)$. If at time $t = t^-$ (before the event) $RD_{X_{n-1}}(t^-)$ has paths with the same weight than the ones in $RD_{X_{n-1}}(t)$ (at event time t) then if the label of the link $X_n - X_{n-1}$ is the same at $t = t^-$ and $t = t$ the result of $\otimes$ will also be the same and therefore at X_n $RD_{X_n}(t) = RD_{X_n}(t^-)$. This means that X_n does not need to change routing decisions and concludes our proof. □

Note that there is a subtle aspect concerning weight preferences. It occurs for paths with $p2patt$ and (bkp, y) weights. For these weights the preference decreases with the number of links of the path, and only one path (the shortest) can be used at a time. Therefore, for this case, if the path in $RD_{X_{n-1}}(t^-)$ is different from the one at $RD_{X_{n-1}}(t)$ (even if it maintains the weight), the control packet has to be forwarded. This is because the result of the $\oplus$ operation with the $X_n - X_{n-1}$ label is different at X_n, that is $RD_{X_n}(t) \neq RD_{X_n}(t^-)$.

Theorem 5.3. *Transient loops caused by control packet inconsistency are contained to one hop and packets loop at most between these two routers.*

Proof. Consider $P_{ij} = \{x_{i0}, x_{i1}, \cdots, x_{ik-1}, x_{ik}, x_n\}$ with $k \in \mathbb{N}^+, 1 \leq j \leq k$ and $1 \leq i \leq$ *number of valid paths to* x_n a path i from AS x_0 to AS x_n. At each x_{ij} along the path a set of i paths to AS x_n exists. Assuming a multi-path routing algorithm then any of these paths might be used. A failure invalidates one or more of these paths, and can cause loops. A loop occurs when one AS (say x_{ij}) has processed the control packet but some of its neighbours (say x_{ij-1}) did not. Then, it can happen that for x_{ij} the new next hop is x_{ij-1} that still has x_{ij} as the next hop forcing the packet to return to x_{ij-1}. This loop is

contained to one hop, and is likely to occur at most once because what happened is that the data packet was sent via the link between x_{ij-1} and x_{ij} at almost the same time that the control packet is travelling in the opposite direction (x_{ij} to x_{ij-1}). Note that more than one data packet can be in this situation and both the processing times and the data packet rates are important to the value of the total number of data packets in the loop. As soon as x_{ij-1} processes the control packet two situations can happen: an alternative path exists and the packet is forwarded to it (and not to x_{ij}), or no path exists and the packet is discarded. Note also that the control packet always arrives because the link between x_{ij-1} and x_{ij} is transporting data packets. □

Theorem 5.4. *Condition: There is at least one available valid path to the destination D during failures. If the condition holds no packet p is lost during the failures*

Proof. Let G be the region graph, and $DG(t)$ the region dynamic graph at time t (with the failed links marked as down). The set of all valid paths rooted at a given AS X is $P(X)$. $DP(X,t)$ is obtained by marking failed paths at time t in $P(X)$. In order to exist at least one valid path to a destination D, D must be a vertex of $DP(X,t)$ (i.e. $D \in V(DP(X,t))$). The condition can then be stated as $D \in V(DP(X,t))$ for a given root node X in $V(DG(t))$. For every packet p flowing from AS A to AS D and being processed at AS X only two reasons exist for discarding it:

1. There is no valid path from X to D at time t. It means that $D \notin V(DP(X,t))$ which contradicts the condition.

2. There is a valid path ($D \in V(DP(X,t))$) but the next hop is exactly the AS where the packet came from (due to the dynamic configuration). Theorem 5.3 guarantees that eventually the packet will not come again.

Therefore, a packet can only be lost if the condition is not met: there is no valid path to the destination. □

5.6 Regions

In terms of scalability the critical aspect is the time complexity to obtain the routing solution. The number of resulting paths is related with the number of ASes and also with

the type of links used (e.g a lot of *p2patt* links turn the network into a mesh instead of a hierarchy and the number of valid paths increases significantly).

One way to reduce the scale is to use regions of ASes [AGA+09]. Regions are sets of ASes with the following two characteristics:

1. ASes in the region must have valid paths to all the other ASes in the region (this is not so drastic because having a provider at a high level solves the problem);

2. Each region must have ASes that connect to every other region via valid paths according to DTIA policies. Moreover, at least one of these ASes (per destination region) must not have providers in the region. This implies that a region must either have a tier-1 AS; or the highest level forms a regional clique without regional providers (such as what is happening currently in Europe), and each of this nodes has providers or peers in the other(s) region(s).

There are no paths spanning multiple regions. The characteristics above assure that a packet to another region can always go directly to it. A packet to the same region never leaves the region (i.e., does not go to another region to come back). As ASes do not know the graph of the other regions they are unable to calculate complete inter-region valid paths. Therefore, inter-region paths combine two paths (one in each region) connected by an inter-region link.

The complete path is divided into a first segment, from the source AS to the border AS, within the first region, and a second segment in the second region. The convergence of the second segment follows from DTIA intra region correctness. Therefore, path convergence depends only on the first segment combined with the inter-region link type. We assume that no information is known about the internal topology of the second region (to contain information and achieve scalability).

The reachable ASes of the other region through each of the border ASes depend both on the type of path followed to arrive at the border AS and on the label of the inter-region link. We can identify three cases:

1. Path arriving in an ascending path (a path that has weight $c2p$) to the border AS and the inter-region link is $c2p$ or $p2patt$. In this case the policy is that every link

departing from the remote border AS opens valid paths. Therefore, all ASes in the destination region are reachable;

2. Path arriving in an ascending path to a border AS connected with an inter-region link of $p2p$ or $p2c$ weights. Only a subset of the ASes of the remote region can be reached. This is because some links departing from the remote border AS might form invalid paths according to the policy rules. For example a $p2p$ inter-region link only gives reachability to the clients of the remote region border AS since a path from a peer cannot be extended to other peers (of any sort) or providers;

3. Path arriving at the border router in a descending path (a path with weight $p2c$). This case is the least interesting since it only allows reaching some ASes with an inter region link of label $p2c$ or $p2patt$. With other link types there are no reachable remote region ASes because the paths are invalid.

Only case 1 assures reachability for any destination AS in the remote region. However, using those cases alone would turn the Internet highly hierarchical (a feature that is disappearing with the peer-to-peer links). Case 2 provides valid paths that might not reach all ASes in the destination region. The problem here is the lack of knowledge of which ASes are reachable, but can be used efficiently. Case 3 has a high chance of having invalid paths or providing only reachability to a subset of ASes and it is the least interesting.

Inter region DTIA only allows paths in cases 1 and 2. In order to use paths in case 2 border ASes exchange a list of ASes they can reach in their region. This reachability list contains the ASes reachable via valid paths assuming that a path reaches the neighbour AS in the ascending direction. The border AS takes in to account the inter-region link type and the regular intra-region policies to produce the list.

In order to maintain DTIA very simple and avoid signalling or information about the other regions, inter-region paths must be calculated and classified in the order of preference according to the above considerations. More precisely:

- the path calculation process begins at the inter-region link and this link has great influence on the final path weight; and

- border ASes calculate the reachable ASes according to the inter-region link label that connects them to each neighbour. This means that different neighbours can receive different reachability lists. For example if the inter-region link is $p2p$ a neighbour that is a provider of the border router receives an empty list of reachable ASes since it does not have a valid path to the remote region using this link. If the neighbour of the border AS is a customer then it receives the clients of the remote region border AS in the list).

The region's characteristic 2 guarantees that, per destination region, at least one border AS exists with an inter-region link with label $c2p$ or $p2patt$. Meaning that it has a provider (or a peer allowing transit traffic) in the remote region. Moreover, this AS has no providers (inside the region) so that it can be reached by all ASes in the region via an ascending path. This is consistent with the current business model in the Internet (no other business relationships assure connectivity to all destinations).

Each AS in a region calculates the valid paths to all border ASes of the region. When it has a packet to a certain region there is a choice amongst the paths to all border ASes for that region. More precisely it is the choice of the first hop. The preference goes to ascending paths (with weight $c2p$) to border ASes with $c2p$ or $p2patt$ links to other regions since they are the ones that can reach all remote ASes. A side effect of such preference is the concentration of traffic at higher hierarchical levels of the Internet. An ideal situation would be the packet to traverse border ASes in situation 2 in its way to those border ASes (in situation 1). Then, at each intermediate border AS, if the destination AS is in its reachability list, the packet is just sent to the other region; if it is not, the border router forwards them to another border AS. Note that this second situation is completely consistent with the current business model: traffic from the border AS clients can go to a remote region via a $p2p$ or $p2c$ links provided that the destination AS is reachable from there. One way to achieve this ideal situation is for the source ASes to choose (since they know the entire region map) amongst the available most preferred paths the ones that transverse more border ASes. This maximizes the possibility of the packet exiting the region earlier, and corresponds to a *hot potato* policy.

Each time a packet arrives at a border AS its reachability list is checked. If the

destination is not in it, the packet is forwarded *"up"* in the hierarchy to another border AS. The process goes on until either the packet is delivered to the other region or a border AS with an inter-region link with label $c2p$ or $p2patt$ is reached.

If no paths with border links $c2p$ or $p2patt$ exist (due to a failure), the source AS chooses one path with a lower preference weight (via a border AS in situation 2). Using one of the lesser preferred paths does not guarantee reachability to the destination AS (only the ASes in the reachability list are reachable), even if a valid path exists somewhere else. This is the price to pay for not having signalling. However, it might only happen in inter-AS failure situations since the existence of a border AS connected via a $c2p$ or $p2patt$ link is a condition to form a region.

It is clear now why situation 3 fits badly in this picture. A packet arriving to a border AS going to a destination that is not part of the reachability list cannot go upwards to another border AS since that is an invalid path. So, to avoid previous signalling we do not consider paths with weight $p2c$ for inter-region paths meaning that paths arriving at the border AS via an $p2c$ are only valid in backup situations.

We modelled these behaviour in an inter-region ordered semigroup. It uses the same set S as the intra-region algebra defined in the previous sections. The $\otimes$ operation for inter region paths is showed in Table 5.3. The preference order $\preceq$ for inter-region paths is showed in Table 5.4.

Table 5.3: DTIA's inter -region $\otimes$ operation

$\otimes$	$\overline{1}$	$p2patt$	$p2p$	$p2pbk$	$c2p$	(bkp, x)
$p2patt$	$p2patt$	$\overline{0}$	$\overline{0}$	$(bkp, 1)$	$\overline{0}$	$(bkp, x+1)$
$p2c$	$(bkp, 1)$	$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$	$(bkp, x+1)$
$p2p$	$p2p$	$\overline{0}$	$\overline{0}$	$(bkp, 1)$	$\overline{0}$	$\overline{0}$
$c2p$	$c2p$	$c2p$	$p2p$	$p2pbk$	$c2p$	$(bkp, x+1)$
$p2pbk$	$p2pbk$	$(bkp, 1)$	$\overline{0}$	$(bkp, 1)$	$(bkp, 1)$	$(bkp, x+1)$

Paths with $c2p$ weight are the most preferred followed by $p2patt$ paths since both assure reachability to any destination AS in the remote region. Next, it comes $p2p$ and $p2pbk$ followed by backup paths. These last paths assure connectivity to just a subset of the destination region ASes.

According to $\otimes$ there are no paths with $p2c$ weight, a path to another region that starts

Table 5.4: DTIA's inter-region order of preference

$c2p = p2patt$
$p2p = p2pbkp$
$(bkp, 1)$
...
(bkp, n)

with an inter-region link with label $p2c$ has the backup weight ($p2c \otimes \overline{1} = (bkp, 1)$) and is only used if no other paths exist. This means that a path starting with an inter-region $p2c$ link always maintains the bkp weight. Also note that paths starting with an inter-region link of type $p2patt$, $p2p$ or $c2p$ are only valid if extended by a $c2p$ link meaning that only ascending paths arriving at such border links are valid. This means that intra-region ASes will only choose from ascending paths to the border ASes.

Like in the intra-region case between bkp paths we use a secondary metric modelled by the lexicographic product:

$$S \overrightarrow{\times} N = (S \times \mathbb{N}, \preceq_{S \overrightarrow{\times} N}, \otimes_{S \overrightarrow{\times} N})$$

where

$$(s_1, n_1) \preceq_{S \overrightarrow{\times} N} (s_2, n_2) \iff s_1 \prec s_2 \vee (s_1 \simeq s_2 \simeq bkp \wedge (n_1 < n_2))$$

and

$$(s_1, n_1) \otimes_{S \overrightarrow{\times} N} (s_2, n_2) = (s_1 \otimes s_2, n_1 + n_2)$$

The backup paths (with weight $(bkp, 1)$) are only used if no other physical paths exist to the other region. This is the same behaviour as for the intra region traffic. However, a useful feature could have existed for inter-region traffic: in failure situations it might happen that a border AS in situation 2 cannot reach a certain AS, but a backup inter-region link could. This situation can only be handled with signalling (something we did not want to consider).

Based on the previous results we prove the correctness of DTIA when routing between two regions.

Proof. We need to check which of the convergence conditions stated in chapter 4 are

verified for the ordered semigroup that models the inter region path calculation and ranking mechanism. We can see that the ordered semigroup $(S, \preceq, \otimes)$ for the inter - region paths is monotone but not strictly monotone since for example $c2p \preceq p2p$ but $p2p \otimes c2p = \overline{0} = p2p \otimes p2p = \overline{0}$. This means that the protocol will not converge in finite time to a routing solution and forwarding loops will occur if *non free* cycles exist in the network.

According to 4.12 in chapter 4 we must then define for each element $s \in S \setminus \overline{0}$ the set $L_s \subset S, L_s = \{l \in S \mid \text{such that } s = l \otimes s\}$ that contains all elements of S that $\otimes$ added to s maintain preference. Cycles in which all links have labels contained in the same L_s will be *non free*.

We have:

- $L_{\overline{1}} = \{\}$ because there is no element $l \in S$ such that $\overline{1} = l \otimes \overline{1}$.
- $L_{p2patt} = \{\}$ because there is no element $l \in S$ such that $p2patt = l \otimes p2patt$.
- $L_{p2p} = \{c2p\}$ Since only $p2p = c2p \otimes p2p$.
- $L_{p2pbk} = \{c2p\}$ Since only $p2pbk = c2p \otimes p2pbk$.
- $L_{c2p} = \{c2p\}$. Since only $c2p = c2p \otimes c2p$.
- $L_{bkp} = \{bkp, p2patt, p2c, c2p, p2pbk\}$. Since for all elements of $s \in S$ except $p2p$, $bkp = l \otimes bkp$.

These sets are concerned only with the ordered semigroup $(S, \preceq, \otimes)$. The total algebra is the lexicographic product $S \overrightarrow{\times} N$ that includes the secondary metric for the bkp paths. According to the definition of $\otimes_{S \overrightarrow{\times} N}$ all sets are equal for the total algebra except the L_{bkp} set that according to $\otimes_{S \overrightarrow{\times} N}$ becomes:

- $L_{bkp} = \{\}$. Since for all elements of $s \in S \overrightarrow{\times} \mathbb{N}$, $(bkp, n) \neq l \otimes (bkp, n)$.

So *non free* cycles occur in cycles that have:

1. All its links labelled $c2p$.

A cycle with all links of type $c2p$ is only possible in a provider customer hierarchy cycle which we assume not to exist in the network since it does not make sense in the current business model of the Internet. This concludes our proof. □

After the proof, some considerations can be made about the reason to invalidate paths to remote regions traversing $p2patt$ links. For example, paths that start with a $c2p$ or $p2patt$ link to the remote region cannot be extended with $p2patt$ links ($p2patt \otimes c2p = \overline{0}$ and $p2patt \otimes p2patt = \overline{0}$).

There is no reason in terms of business model to invalidate such paths. However, allowing them would greatly increase the possible *non free* cycles. Changing $p2patt \otimes c2p$ to the result $c2p$ and $p2patt \otimes p2patt$ to $p2patt$ would cause set L_{p2patt} to become $\{p2patt, c2p\}$ and set L_{c2p} would also be $\{c2p, p2patt\}$ meaning that cycles having all its links labelled $c2p$ or $p2patt$ would be non free. This would mean that a simple cycle like the one in figure 5.4 would be *non-free*. In this example a forwarding loop can occur between AS B and AS C because paths $B - A$; $B - C - A$; $C - A$ and $C - B - A$ all have equivalent preference. Since forbidding such cycles in the network does not seem reasonable we opted to invalidate $p2patt \otimes c2p = \overline{0}$ and $p2patt \otimes p2patt = \overline{0}$.

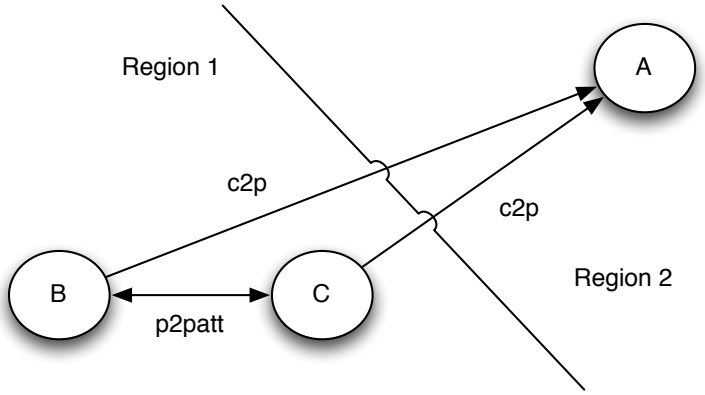

Figure 5.4: *Non free* cycle example inter-region

Inter-region DTIA relies on all ASes being reachable in the remote region if either a $c2p$ or a $p2patt$ inter-region link exists. The following theorem has to be proven.

Theorem 5.5. *Assuming that the preference rules specified in Table 5.4 are followed, if either a c2p or a p2patt inter-region link exists, then, the DTIA routing protocol converges*

to a set of loop-free paths that reach the destination AS.

Proof. The proof follows from the policy rules used in intra-region DTIA, namely by the $\otimes$ operation defined in table 5.1. If a border AS of the remote region is reached either via a *c2p* or a *p2patt* link, then all paths in the remote region can be extended by those links without becoming invalid (the *p2patt* and *c2p* lines in table 5.1 never have $\overline{0}$ weight). If there are no invalid paths all ASes are reachable from the origin region via such paths. □

5.6.1 Failures between regions

Regarding the handling of the failures, the inter-region case is quite similar to the intra-region one with two differences related with the inter-region link. The first occurs for the case of inter-region links of backup type. As stated above they are used only in the case of all the other links are down. The second happens when an inter-region link of type *c2p* or *p2patt* fails (and it is even worse if it is the more preferred one). In this case it might happen that certain ASes of the destination region become unreachable. Note that this is a serious failure and given the current Internet there are not many records of failures at this level.

5.7 Deployment

The deployment process cannot be based on a synchronized change of the entire world at the same time. Also, as stated in the introduction, a new system will not have all the flexibility of BGP, but will have other features. Most likely a situation of "*early adaptor advantages*" will not happen here. For the change to happen there must be an agreement on the advantages of the new features and a willingness to change. We assume that (a) the network graph can evolve from the effort of RIPE, (b) there is a mapping service to know AS identifiers from prefixes, and (c) the mapping service must also provide the prefixes belonging to a certain AS.

The interworking with BGP-4 is a major requirement. ASes running BGP-4 stay as they are. The new system has to be deployed from bottom to up always forming convex areas. This means that an AS can only change if all its customers have changed. Regions start to exist with graphs containing a few ASes, and assume that the rest of the world is

in another region. We show in [AGA+09] that a simple approach can be used to connect DTIA *islands* to the rest of the Internet: DTIA-BGP can be handled by a mapping of all IP addresses outside the region to a default AS *id* handled by certain border routers. The border routers know which ASes inside the region are reachable. Using the mapping service they advertise in BGP to the exterior the respective prefixes. Peer-to-peer traffic between DTIA world and BGP world is handled by direct links and is treated as an exception in a step prior to the mapping service.

A similar process can be designed to deploy a new version of DTIA with a different algebra.

Chapter 6

DTIA Traffic engineering protocol

6.1 Introduction

DTIA features a multipath routing environment without forwarding loops using simple destination based hop by hop forwarding that allows traffic to be split arbitrarily amongst the different paths. In such an architecture a finer control over traffic is possible and traffic control can be achieved without changing routing configurations and without influencing route dissemination. This provides a new way to address the inter-domain TE problem: define the correct traffic distribution to achieve the TE goal instead of defining the correct route attributes that would lead to a routing state that produces the desired traffic distribution.

In this chapter we explore these DTIA's characteristics to perform TE: multipath routing, and the ability to separate traffic control from the routing state. The intention is to assess what can be achieved by using them, keeping in mind that the Internet imposes limitations in terms of available information and AS coordination. Therefore, we follow a very lightweight approach to design a TE protocol that only uses: DTIA characteristics; locally available information at each AS; and minimal signalling consisting in a simple control packet. We then assess the amount of traffic control that is possible in such a constrained approach.

Our protocol performs what is known as online TE. It starts when an AS is receiving too much traffic from a certain source. Using DTIA's routing and traffic information from the links directly connected to it, the AS can identify another AS that can change its

behaviour in terms of traffic distribution. The AS connected to the overloaded link issues requests for cooperation to specific remote ASes. Cooperating ASes change their traffic distribution according to the feedback received from the initiating AS in order to avoid congestion in the network. The incentive to a cooperating AS is to increase the possibility that traffic leaving from it will avoid congested paths. The protocol acts similarly to a feedback mechanism that increases the cost of a path when one of its possible segments is becoming congested. We tested two different variations of the protocol in order to analyse the trade-off between performance and complexity. In the next section we show how remote routing information can be obtained in DTIA. Then in the remaining sections of this chapter the DTIA TE protocol is presented in both versions.

6.1.1 Inferring remote routing information in DTIA

In DTIA all ASes have the AS level map of the network and calculate paths according to the $\otimes$ operation. Consider a valid path P from X_1 to X_n as $P = X_1, X_2, X_3, \ldots, X_{n-1}, X_n$ and denote $L(X_i, X_j)$ the label of the link in the X_i to Xj direction. For example, if X_j is a provider of X_i then $L(X_i, X_j) = c2p$ and $L(X_j, X_i) = p2c$. The weight of the path P is $W(P) = L(X_1, X_2) \otimes (\cdots \otimes (L(X_{n-2}, X_{n-1}) \otimes (L(X_{n-1}, X_n) \otimes \overline{1}))$.

AS X_1 can apply $\otimes$ to the reverse path: $rP = X_n, X_{n-1}, \ldots, X_3, X_2, X_1$, and calculate the weight of this path in the X_n,X_1 direction $W(rP) = L(X_n, X_{n-1}) \otimes (\cdots \otimes (L(X_3, X_2) \otimes (L(X_2, X_1)) \otimes \overline{1}))$. This means that by just knowing the map and the valid paths towards another AS X_n, a certain AS X_1 can calculate the paths from AS X_n toward itself by just reverting the outgoing paths.

What remains to be seen is whether more valid paths exist from an AS X_n to AS X_1 other than the reverted valid paths from X_1 to X_n.

In DTIA's ordered semigroup the following condition holds:

Condition 6.1. *Consider two ASes X_1 and X_n. All reverted paths of valid paths from X_1 to X_n are valid paths from X_n to X_1. The only other possible paths from X_n to X_1 have weight bkp.*

This means that X_1 can infer **all** valid paths that reach it from any given AS with a weight different from *bkp*. Or in other words, it knows the paths that remote ASes have to

reach them in failure free networks. The proof is presented next. It follows the reasoning that any invalid path from X_1 to X_n cannot be a valid path from X_n to X_1 except if it is a path with *bkp* weight.

Proof. In terms of the ordered semi-group that models DTIA this means that for every path P if $W(P) = \overline{0}$ then $W(rP) = \overline{0}$. We prove this by contradiction. Consider $P = X_1, X_2, X_3, \ldots, X_{n-1}, X_n$ and denote $W(X_i, X_n)$ as the weight of the sub-path between X_i and X_n, (between hop i and the destination). We have $W(P) = \overline{0}$ if and only if for some term i $L(X_{i-1}, X_i) \otimes W(X_i, X_n)) = \overline{0}$ with $i \in \{1, 2, 3, \ldots, n\}$. That is, if one of the partial path weights extended with the next label in the path is invalid. Following the same reasoning, the reverse path is valid for the same term i if and only if $W(X_n, X_i) \otimes L(X_i, X_{i-1}) \neq \overline{0}$ with $i \in \{1, 2, 3, \ldots, n\}$. We then examine in Table 5.1 in chapter 5 each of the invalid results of $\otimes$. There are six cases. We examine each one and find the reverse weight for each case. We exemplify for the first case:

$W(1) = p2c \otimes p2p = \overline{0}$. The reverse weight is $rW(1) = p2p \oplus c2p$ corresponding to a path where we have a segment that results in weight $c2p$ followed by a $p2p$ link in the direction of the source. At this point the result is also $\overline{0}$ meaning that the reverse paths of paths containing a segment with result $p2c \otimes p2p = \overline{0}$ are invalidated at a point where the result $p2p \oplus c2p = \overline{0}$ happens.

The application to the other five cases is straightforward:

1. $W(1) = p2c \otimes p2p = \overline{0}$. The reverse path weight is $rW(1) = p2p \oplus c2p = \overline{0}$
2. $W(2) = p2p \otimes p2p = \overline{0}$. The reverse path weight is $rW(2) = p2p \oplus p2p = \overline{0}$
3. $W(3) = p2c \otimes c2p = \overline{0}$. The reverse path weight is $rW(3) = p2c \oplus c2p = \overline{0}$
4. $W(4) = p2p \otimes c2p = \overline{0}$. The reverse path weight is $rW(4) = p2c \oplus p2p = \overline{0}$
5. $W(5) = p2bkp \otimes p2p = \overline{0}$. The reverse path weight is $rW(5) = p2p \oplus p2pbk = bkp$
6. $W(6) = p2p \otimes bkp = \overline{0}$. The reverse path weight is $rW(6) = bkp \oplus p2p$. Since *bkp* is not a possible link label we have to see all possible versions of $rW(6)$ replacing *bkp* by the $\otimes$ operations that result in *bkp* in the reverse direction. We have:
   (a) $p2patt \otimes p2pbk$. With $rW(6) = p2pbk \otimes (p2patt \otimes p2p) = bkp$

(b) $p2c \otimes p2pbk$. With $rW(6) = p2pbk \otimes (c2p \otimes p2p) = bkp$

(c) $p2pbk \otimes p2pbk$. With $rW(6) = p2pbk \otimes (p2pbk \otimes p2p) = \overline{0}$

(d) $p2pbk \otimes p2patt$. With $rW(6) = p2patt \otimes (p2pbk \otimes p2p) = \overline{0}$

(e) $p2pbk \otimes c2p$. With $rW(6) = p2c \otimes (p2pbk \otimes p2p) = \overline{0}$

Therefore, for every invalid path in the forward direction, the reverse path is also invalid or a backup-only path and this concludes our proof. □

6.2 DTIA traffic engineering protocol

DTIA routing is based on business policies. It provides multipath with routing stability and a mean to infer routing information of the other ASes. There is room now to perform TE by managing the traffic distribution by the available paths. DTIA intrinsically separates traffic control from route dissemination. The set of routes is stable and instead of trying to fine tune routes TE can be performed by finding the correct traffic distribution between the multiple paths. This is different from the traditional research since there is a stable set of routes without the need for tunnels or Label Switched Paths. Forwarding is still hop by hop and there is no global state on paths. Although this provides less control than tunnelling, it does not require signalling or state across ASes and is more resilient to failures. Taking advantage of this characteristics we focused on a network wide congestion avoidance algorithm based on link usage feedback. To maintain scalability and AS independence we only use local available traffic information and a unique feedback signalling packet.

The protocol works like a feedback mechanism that uses a signalling packet *congestion feedback packet (CFP)* that is sent by ASes that detect a congestion in one of their incoming links. It works by informing remote ASes about congested local links. Remote ASes can then redistribute traffic to other preferred paths which are more likely to avoid the congestion. The incentive to the remote AS is to forward its traffic through less congested paths, therefore reducing delay and packet loss. The redistribution of traffic will result in better traffic distribution depending on the number of cooperating ASes. The protocol has three phases: congestion detection, CFP distribution and traffic redistribution. In the

remaining three sections of the chapter we describe each one of them in more detail.

6.2.1 Congestion detection

Consider that one AS, AS_{IN}, has a set of incoming links M. The load on each link m_i is monitored to see if a threshold value, ψ_{mi}, is crossed. If the threshold is passed AS_{IN} identifies the source-destination pair that is the the biggest contributor to the link load and sends a CFP packet to it. The traffic load for a source destination pair $s-d$ is given by λ_{s-d}/C_{mi} where λ_{s-d} is the load arriving from source s to destination d and C_{mi} is the total load capacity of link m_i.

6.2.2 CFP distribution

We actually implemented two different versions of the algorithm concerning the CFP packet distribution: pure feedback and selective feedback.

In the pure feedback version, AS_{IN} identifies the largest contributor, AS_{Os}, and sends to the upstream neighbour connected through m_i a CFP packet that contains: the AS number of AS_{Os}, the AS number of the destination AS_{Od}, the AS number of the neighbour connected through m_i, denoted by ν_{mi}, and the set of neighbours of AS_{IN} that also route traffic from AS_{Os} to AS_{IN}, which is denoted by NOs_{IN}. NOs_{IN} contains the alternative ASes that can be used to route traffic from AS_{Os} to AS_{IN} avoiding the congested link, m_i.

When the AS ν_{mi} receives the CFP packet it checks if it has alternative paths to AS_{IN} with the same weight and changes the traffic distribution to avoid the congested link. It then finds its own set of neighbours that also route traffic from AS_{Os} ($NOs_{\nu_{mi}}$), and a CFP is sent to ASes in $NOs_{\nu_{mi}}$. The process is repeated by every AS that receives a CFP until the AS_{Os} is reached. The simple topology in Figure 6.1 serves as an example.

Consider that AS C is the AS_{IN}, that at a given time t the link C-D becomes congested, and that the flow with more packets is G-C. A CFD packet is sent to AS D containing $AS_{Os} = G$, $AS_{Od} = C$, $\nu_{mi} = D$ and $N_{Oi} = F, E$. Since AS D has two paths of $c2p$ weight to C it changes the traffic distribution sending more traffic through AS F. It then sends CFD packets to NOs_D (the set of neighbours sending traffic from origin G) that contains

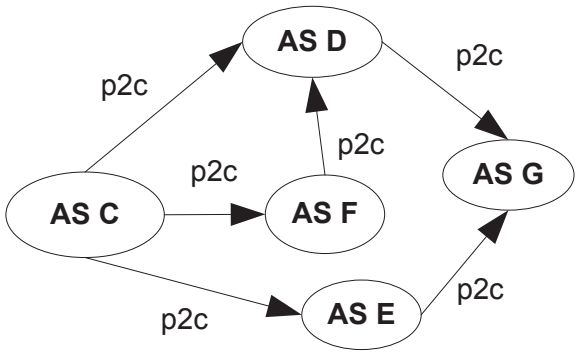

Figure 6.1: Simple topology

only AS G. Since G is the origin, no more CFDs are sent. G checks if it can reach C through more than one path, and in this particular case it increases the amount of traffic through the G-E-C path, stopping the process.

In the selective feedback version, we use the possibility to infer the routing information of AS_{Os} (by computing the weights of the reverse paths) to send the CFP packet directly to the source of traffic.

We use the inferred paths from AS_{Os} to AS_{IN}, let F be the set of first hops in those paths. For every member f_i we can know, by applying the same process, the paths from f_i to AS_{IN}. We can then repeat the process until we reach AS_{IN}. A CFP packet is sent to all f_i ASes that have alternative paths for the destination AS_{Od}. This uses DTIA remote routing inference capabilities to try to reduce the number of CFP packets. We use Figure 6.1 again as an example with the link C-D congested due to the flow G-C. In this case AS C will infer the paths from AS G to itself. There are three paths with weights $c2p$ (G-D-C,G-D-F-C and G-E-F) where the first two are reduced to one since the first hop is the same (AS D). So a CFP packet is sent to AS G. AS C then checks the paths from D and E to itself (D and E are the first hops of paths from G to C). AS E has no alternatives and therefore a CFD message is not sent from C to E and this part of the process stops. AS D has alternatives and therefore a CFD packet is sent to it by C. Next, D's first hops to C (F and C) are evaluated. Since C is the destination, only F is checked for paths. AS F has no alternative paths and therefore CFP packets are not sent. Since F is directly connected to C, the algorithm stops.

For both versions, when the link is no longer saturated and the total load from AS_{Os} decreases a RESET packet is sent to return to the previous default traffic distribution.

The final part of the protocol is the redistribution of traffic between possible paths after the reception of a CFP packet.

6.2.3 Traffic redistribution

The reception of a CFP packet informs an AS that a congestion is occurring in the link m_i between AS_{IN} and ν_{mi}, and that the flow with destination AS_{Od} is the higher contributor to the congestion. To evaluate if it has alternative paths it checks the forwarding table, both for AS_{Od} and ν_{mi}. If it finds first hops that only reach AS_{Od} and not ν_{mi} it knows that those are capable of reaching AS_{Od} avoiding the congested link. The paths starting with those first hops should receive more traffic since they assure delivery avoiding the congested link. Paths to AS_{Od} and ν_{mi} starting with the same first hop can diverge along the way and also avoid link m_i. However, since DTIA has no explicit end-to-end paths, the path followed after the first hop depends on the routing decisions of the other ASes and therefore, avoiding m_i is not guaranteed in this case. The protocol must then distribute more traffic to the links of the first kind than to these.

In order to achieve that distribution we use a simple exponential penalty based scheme inspired on [Xu08]. The distribution of traffic is performed by assigning a higher penalty value to paths that reach both AS_{Od} and ν_{mi} and a smaller penalty to the ones that only reach AS_{Od}. Considering the multiple paths between i and j indexed by k, the traffic distribution ratio is

$$\frac{X_k^{(i,j)}}{X^{(i,j)}}$$

where $X_k^{(i,j)}$ is the amount of traffic in path k and $X^{(i,j)}$ the total traffic through all paths. The distribution for path k is given by:

$$\frac{e^{-pe_{k^{(i,j)}}}}{\Sigma_n e^{-pe_{n^{(i,j)}}}} \tag{6.1}$$

with n being the n paths available and $pe_{i^{(i,j)}}$ the penalty for path i.

This means that the fraction of traffic from i to j distributed for a path is inversely proportional to the exponential of its penalty value. Or in other words, it decreases exponentially with the increase of the penalty value. The result is an exponential decaying

rate of traffic in the links that can be responsible for the congestion. Results of the implementation of DTIA and its TE protocol are discussed in the next chapter.

Chapter 7

DTIA results

7.1 Introdution

This chapter contains various types of experiments. In order to perform them the DTIA architecture had to be implemented. We followed two different approaches according to the specific experiments: one was to implement DTIA on the discrete event simulator *ns2* and use an implementation of BGP for ns2 called BGP++ [BGP] for comparative experiments between DTIA and BGP; the other approach was to implement an emulator in JAVA were the calculation of the routing solution and the building of the forwarding table were emulated in terms of control procedure instead of a full discrete event simulation. The reason to build the emulator was the need to study DTIA in terms of scalability, bigger topologies are needed and they are not feasible using *ns2* due to its limitations in terms of computational resources. Both implementations (*ns2* and JAVA emulator) calculate the path weights of a network topology and rank them according to preference, details on the algorithm followed to implement DTIA both in ns2 and in the JAVA emulator can be found in Appendix A. The implementation in the discrete event simulator also simulates the failure handling algorithm and the inter-region operation. For some comparative experiments we also implemented an emulator of BGP in JAVA that calculates the paths obtained in BGP following the common business policies on a given network topology. The aspects of multihoming and multipath are presented in the next section together with the description of scalability experiments. All these aspects are addressed in terms of procedural analysis due to the great dependency on concrete topologies that prevents

any specific experiment to be clearly conclusive. This chapter also covers the handling of failures and the use of regions. Finally, the last part of the chapter describes the experiments performed with the TE protocol. The TE protocol works on top of DTIA and was also implemented in the *ns2* simulator. The implementation follows the algorithms described in chapter 6 and implementation details can be found in [Sil10]. Some minor changes were made to the previous *ns2* code for DTIA regarding the need to infer remote paths [Sil10].

7.2 DTIA's scalabilty

In terms of scalability we intended to assess the necessary time to calculate the path weights and build the forwarding table by applying the preference order. The purpose is to test the time complexity of the DTIA algorithm. A second aspect was to check the size of the stored data. DTIA stores two tables: one containing all path weights for each destination X, called $Pr(X)$, and another, the forwarding table that contains only the most preferred paths for each destination X, called $FH(X)$. We used the emulator for these experiments. It receives an AS level map and calculates the path weights, as a DTIA router would do, then it minimizes the preference and stores in $FH(X)$ the more preferred paths with different first hops. Appendix A contains a brief description of the implemented algorithm. Further details can be found in [Gan09].

We used a topology with 11,335 ASes and over 21,000 links obtained from the CAIDA AS relationships Data research project [CAI], comprising 76 countries (the RIPE region). It is a large region that was chosen in order to provide insight about the upper limits of DTIA algorithm for the current Internet. The calculations were performed by a machine with an Intel Q6600 processor and 8 GB of RAM.

ASes have different characteristics according to their position in the customer provider hierarchy. Instead of assigning ASes to *tiers* we decided to classify them by the number of their neighbours. "*Higher level*" ASes usually have more links than smaller client ASes. We divided the 11,335 ASes into five groups (shown in the *x-axis* of Fig. 7.1). The Figure plots the number of ASes in the topology that falls in each of the five groups.

We can see that (as expected in an Internet like network) the number of ASes with

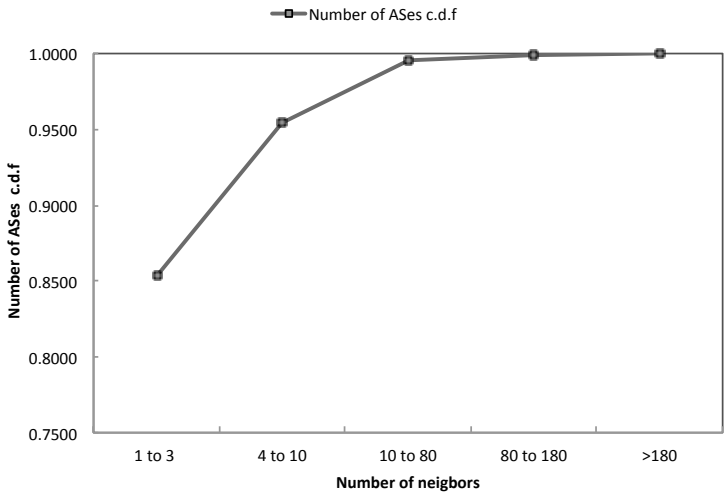

Figure 7.1: Cumulative distribution function of the number of ASes per group in the used topology

less than 4 neighbours accounts for 85.34 % of the ASes in the topology. About 95.4% of the Ases have less than 11 neighbours, only 0.4% have more than 80 neighbours and only 0.25% of the ASes in the topology have more than 180 neighbours.

In Figure 7.2 we plot the average $FH(X)$ size and average processing time for each of the five groups.

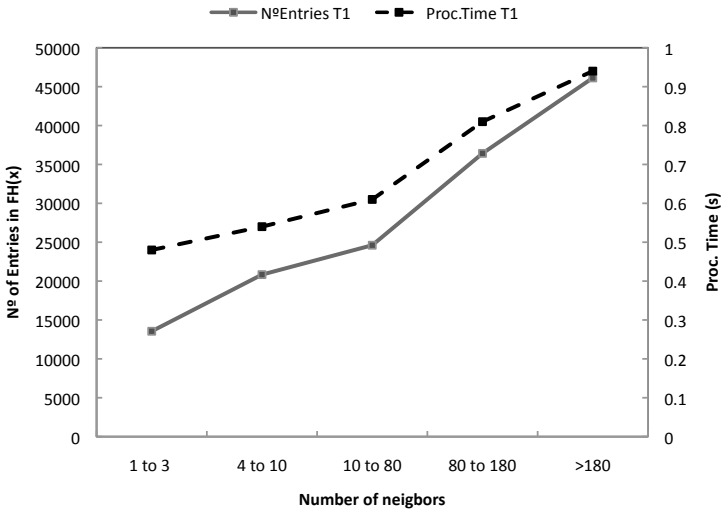

Figure 7.2: Number of $FH(X)$ entries and processing time

Despite the large number of ASes, the largest forwarding table size is 46 127 entries long for the >180 group. The highest measured processing time in an AS was 1.03s (also

for an AS in this group) and the highest average processing time per group is below 1s which is quite reasonable. This group has only 0.25% of the ASes of the region. The three lower groups (ASes with at most 80 neighbors) account for 99.6% of the total ASes and the greatest table has 24 611 entries. The two lower groups accounting for 95.4% of the ASes have a table with a mean size of 20 833 entries.

Apart from the $FH(X)$ table, DTIA needs to store $Pr(X)$. $Pr(X)$ contains the first hops and the value of all paths for a given destination. Some comparison with BGP can still be made despite the fact that the entries represent different things and the $Pr(X)$ table contains more information than, for instance, the entries in the routing information base (RIB) of a BGP router.

The largest number of entries for the above topology was 79 777 entries, considering both the forwarding table $FH(X)$ and the remaining paths in $Pr(X)$. We used the publicly available data from the RIPE Routing Information Service (RIS) [RIP] to obtain the size of a RIB table in BGP for the same topology. We used the data from one of the RIS locations (AS 12654) that has fifteen routers in different locations having more than 600 peers. Its routing table gives a large view of the global Routing Table in the Internet. The full RIB had 3 103 335 routes and the forward information base (FIB) table had 305 420 routes. We then eliminated all routes for prefixes that do not belong to our region (11 335 ASes). We obtained 1 545 985 and 64 345 entries for the RIB and FIB respectively. Although they are not directly comparable, it shows that DTIA's amount of stored information is reasonable.

Regarding the processing time we observe an increase of roughly 50% in the processing time between the less connected group of ASes and the densely connected group. The evolution of the processing time follows the increase of the number of entries in the forwarding table and both grow linearly with the number of neighbours of the AS. The critical scalability factor for DTIA seems to be more the number of edges of the network than the number of nodes. Considering S the set of link labels and path weights, E the set of edges (links) in the networks graph and V the set of vertices (nodes). The worst time complexity of the algorithm is $O(|S| \cdot |E| \cdot |V|)$. However, since the vast majority of paths in DTIA are likely to be not Hamiltonian (there are six combinations of links

that result in $\overline{0}$) the real time complexity is more influenced by the $|S|\,.\,|E|$ term than by the $|V|$ value because for the majority of paths the number of vertices is well below the cardinality of V.

7.2.1 Multihoming support

Multihoming refers to the connection of one AS to more than one provider. This can be used for fault tolerance situations or to perform load balancing [BGT04].

Providing fault tolerance is the simplest case: ASes have to announce their prefixes through all their providers and use local preference to define one AS as the default one maintaining the others as backups. Even this simple form of multihoming causes difficulties in BGP. Providers can no longer aggregate prefixes because of the obvious impact on the Routing Table growth [BGT04].

Multihoming also introduces new possible paths and ASes might want to perform load balancing using these paths [BGT04]. It is a primitive way to perform load balance as it is based on separation of addresses with an even greater impact on the routing table growth. ASes have to subdivide their prefixes into two or more sub-prefixes, due to the single path nature of BGP and the longest prefix match rule. To maintain connectivity in case of failure they also have to additionally announce the full aggregate through all providers. According to [BGT04] 20%-25% of the routing entries were due to load balancing, which was at the time the fastest growing cause for the Routing Table.

BGP allows taking advantage of multihoming, but it is only exploited locally. More specifically, one AS can receive advertisements for a certain prefix with different paths. It might use both but can only advertise one. So, beyond this AS this particular multihoming is not known.

If one could announce more than one route for a given destination load balancing would be possible without prefix manipulations and new paths would be available to more ASes. However, several problems arise for BGP. First, advertising and installing multiple paths introduces new difficulties in ensuring overall convergence, and coordination between ASes would be necessary to ensure that routing decisions are coherent. Secondly, the possible gain in routing table growth due to better prefix aggregation would be impaired by the

growth in the number of installed paths. Finally, the number of messages exchanged will increase with the newly advertised routes.

DTIA's approach is simpler and more powerful. DTIA's multipath routing capabilities can take advantage of the various paths for load balancing or other traffic manipulations, transparently. All paths are known to all ASes and performing load balancing has no consequences on configuration. It is up to a higher layer protocol (above routing) to take advantage of them. In terms of scalability increasing the number of connections between ASes increases the size of the tables in DTIA but there are no extra messages being exchanged and in case of failures the probability that an event is contained locally even increases. Also, convergence is still guaranteed.

We performed a simple experiment to understand the trade off involved in taking advantage of multihoming. We emulated a simplified scenario in which BGP is extended to export more than the single best path.

The experiment uses a small but real topology obtained by selecting ASes from the CAIDA AS relationships data research project [CAI]. The topology has 54 ASes and 517 links and it is built by a subset of stub ASes from Portugal, and the set of transit ASes they use, up to *tier-1* (assumed as ASes without providers). It includes ten *tier-1* ASes at the top, and a set of lower tier transit ASes connected by $p2c$ links. This subgraph of the RIPE region is densely connected, it has a high degree of multihoming, and it contains a great number of $p2p$ links that connect ASes from multiple tiers. The connections reported in the CAIDA topology were cross checked with the information in the RIPE registry and missing links added. Figure 7.3 illustrates the topology as seen in the ns2 nam (network animator).

This topology was also used for all other experiments performed with the ns2 simulator. Cyan links represent peering links and gray links represent provider to customer links with the higher (in the picture) nodes being the providers.

The BGP emulator implements the decision process of BGP according to the *common polices* [CR05]. It calculates the routes by emulating route announcements to neighbours according to the policy and choosing the n best paths to install and announce. Paths are considered different if their first hop is different. We used only one prefix per AS. This

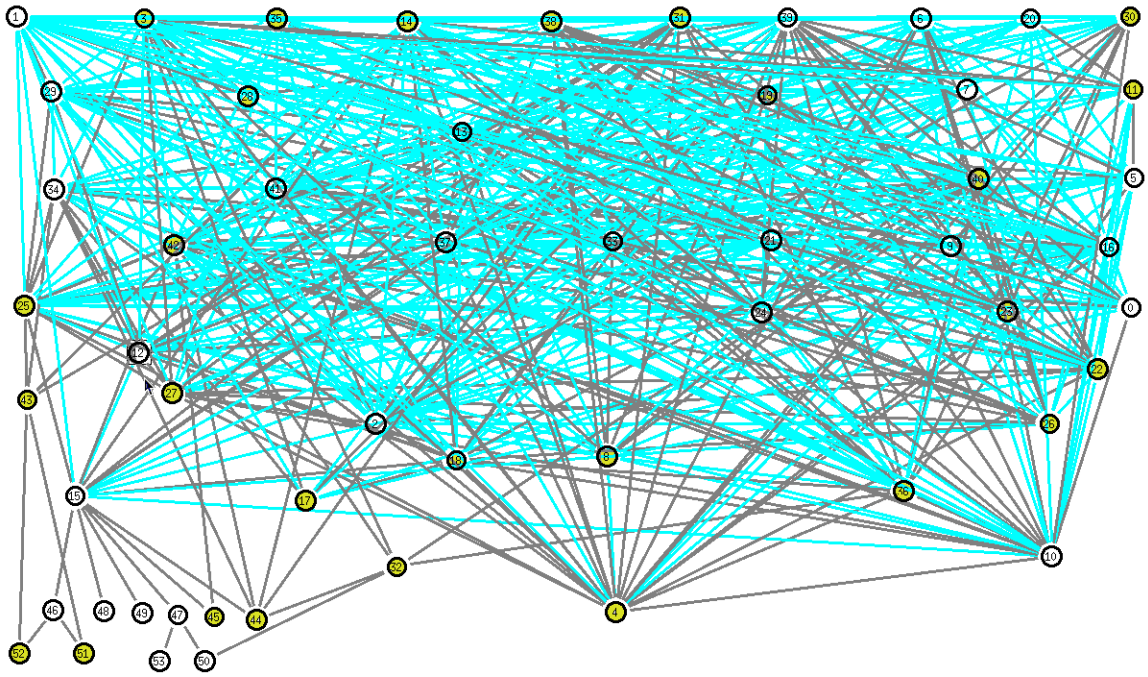

Figure 7.3: Nam visualization of the used topology

simplification greatly reduces the routing table for BGP since typically ASes announce more than one prefix. The purpose was to make it more comparable with DTIA (which routes by AS numbers instead of prefixes). The calculated routing tables represent a lower bound on the BGP routing table size.

We calculated the routes for regular BGP (best path only) and for BGP advertising up to 16 paths. Fig. 7.4 shows the number of route table entries for the various cases. The *common polices* were used to emulate the route export process (as it is not known, obviously). We divided the ASes into six groups according to the number of neighbours, leaving the *tier-1* ASes in a separate group. The *tier-1* ASes form a clique with *p2p* connections between them. This implies that at least one valid path exists between every pair of destinations. Using only one prefix per AS and emulating the decision process using similar policies to the ones of DTIA it is expected that when more paths are announced in BGP both systems will behave similarly. For BGP without multipath the average routing table size was, as expected, 54. As we increase the number of announced paths, n, the number of entries increases, but there is a limit that is dependent of the route diversity of the specific topology.

For DTIA, $FH(X)$ stores paths with different first hops. For ASes with few neighbours

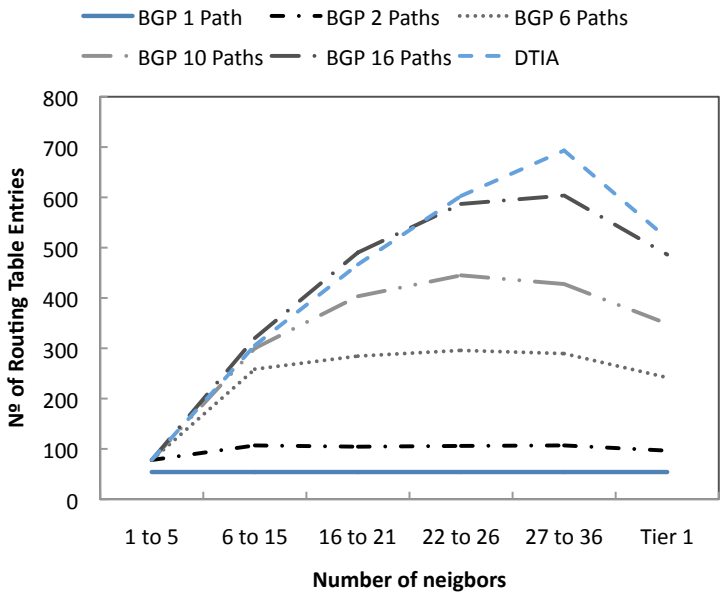

Figure 7.4: Routing table entries for BGP multipath and DTIA

the number of entries is small, despite the number n. In BGP these ASes have a similar reduction on the number of entries, for the same reason.

More connected ASes are more sensitive to increases of the n and we can see in Fig. 7.4 that for the BGP case the number of entries also increases. With $n = 16$, we have a similar number of entries as for DTIA (between 600 and 700 for 54 ASes) and this is probably the limit on route diversity for this particular topology. Although similar in scale in terms of table entries, the other discussed BGP drawbacks still occur (overall convergence, poor scaling due to increased number of exchanged messages etc.). Moreover, DTIA provides multipath natively, does not have churn problems and ensures convergence while using the extra paths to improve the behaviour under failures.

In Fig. 7.4 we also observe that for $n \geq 6$ the *tier-1* ASes have smaller tables than the ASes of the previous groups. Also some of the highly connected ASes that have only one provider (and a lot of clients and peers) have slightly smaller tables. The reason is the following: the link type from which more paths can be received using the *common polices* is the $p2c$ type (all routes can be announced to a client). Therefore, since *tier-1* ASes have no providers they receive fewer routes when using a high number of paths.

The same happens in DTIA and for the same reasons, since the $p2c \otimes X \ \forall X \in S$ the

operation with more results different from $\overline{0}$.

7.2.2 Failure propagation / churn

One of the biggest problems of BGP that impacts also its scalability is the high churn rate and slow convergence after routing events. The increase of multihoming either at stub level or in the mid-tier section of the Internet topology worsens the problem. In [EKD10] the effect of the increased multihoming degree (mean number of providers per AS) in several zones of the topology was studied. An increase of 1.6 in churn was measured following an increase of 3 in the multihoming degree. This is a strong difference to DTIA. In DTIA, increasing the multihoming degree leads to a decrease of the churn rate measured in terms of created control packets. Failures are more likely to be contained in heavily multihomed topologies because it is more likely to have several alternative routes with the same signature to most destinations. Multihoming actually improves DTIAs convergence time and reduces churn after a routing event.

We conducted an experiment using the *ns2* simulator to compare the convergence and churn rate of DTIA and BGP to evaluate DTIA convergence after a routing event.

We used the topology of figure 7.3. This topology is densely connected and has a high degree of multihoming. Figure 7.5 plots the cumulative distribution function of the node's degree for the topology. The node degree is the number of connections to other nodes.

We can see by the graph that this topology is much more densely connected than the 11335 AS topology used in the scalability experiments. Here 60% of the ASes are connected to more than 20 other nodes which means they are connected to at least 37% of the network nodes. This is because this topology is centered in higher ASes in the provider customer hierarchy and only contains the smaller ASes from Portugal. While in the larger topology we have all small ASes from Europe which constitute a vast majority.

Sixty six single links failures were produced, and we measured the number of affected ASes, both for DTIA and BGP.

Figure 7.6 shows the cumulative percentage of ASes that receive a control packet in DTIA or that receive a route withdraw packet in BGP.

DTIA reduces the reach of the churn: 70% of the failures affected less than 5 ASes.

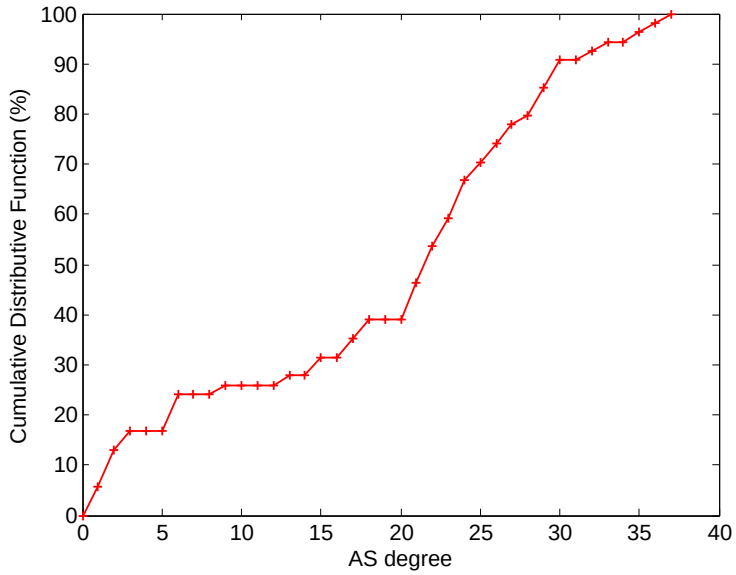

Figure 7.5: Cumulative distributive function (%) of the AS degree

80% of the failures affected less than 15 ASes and only 10 % affected more than 35 ASes. BGP cannot restrict churn so well. 5 or less ASes were affected only for 8% of the failures. 25 % of the failures affected 15 or less. This experiment indicates that DTIA greatly reduces the number of ASes that know about a failure and consequently reduces churn.

7.2.3 DTIA with two regions

To test the effects of using regions we divided the topology of figure 7.3 in to two regions following the characteristics defined in chapter 5 section 5.7 for a region. Nodes in yellow in the figure were part of region 0, R_0, and the nodes in white where part of region 1, R_1.

To measure the reduction in the stored information in DTIA we measured the number of entries in the nodes when the entire topology is a single region and when it is divided into two regions. We considered the total number of entries $FH(X)$ plus $Pr(X)$. Figure 7.7 plots the cumulative distribution function of the number of entries in the tables.

It is clear that the splitting of the topology into two regions reduces the amount of entries. This is due to the fact that details from remote regions are hidden, even if border routers receive lists of reachable remote destination ASes. For the case of a single region 60% of the ASes have more than 600 entries. These values drop to about 200 entries for

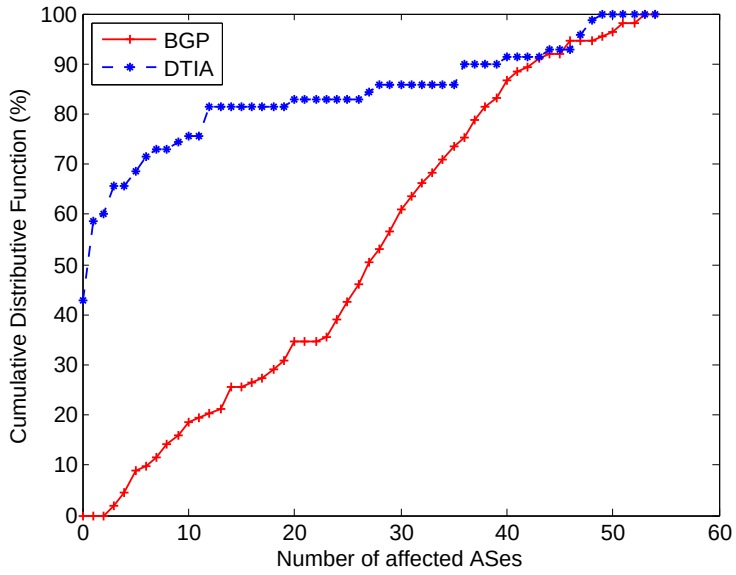

Figure 7.6: Cumulative distributive function (%) of affected ASes after a single link failure

the two regions' case.

A side effect of dividing the topology into regions is the possible increase in path lengths due to hierarchical routing. To asses this we sent probe packets through all the inter-region paths and compared the path lengths for a single region and for two regions. We observed that 92% of the path lengths did not change; 6% of the paths' length increased an average of 2 hops; and 2% of the paths' length actually reduced an average of 1 hop. The reason for such resemblance is due to the strong degree of multihoming. Multi-region DTIA does not use $p2c$ paths, whereas single-region DTIA prefers $p2c$ paths. Therefore 2% of the paths through $c2p$ or $p2p$ links were shorter than the previously used $p2c$ paths.

Finally, we tested the effect of an isolated failure. Figure 7.8 shows the cumulative percentage of ASes that receive a control packet after a failure, we tested four different cases: for BGP and for DTIA with one region (DTIA 1) after a link failure; and DTIA with two regions for a failed intra-region link (DTIA 2i) and for a failed inter-region link (DTIA 2b). The results show that the subdivision into regions further contributes to contain failure advertisements.

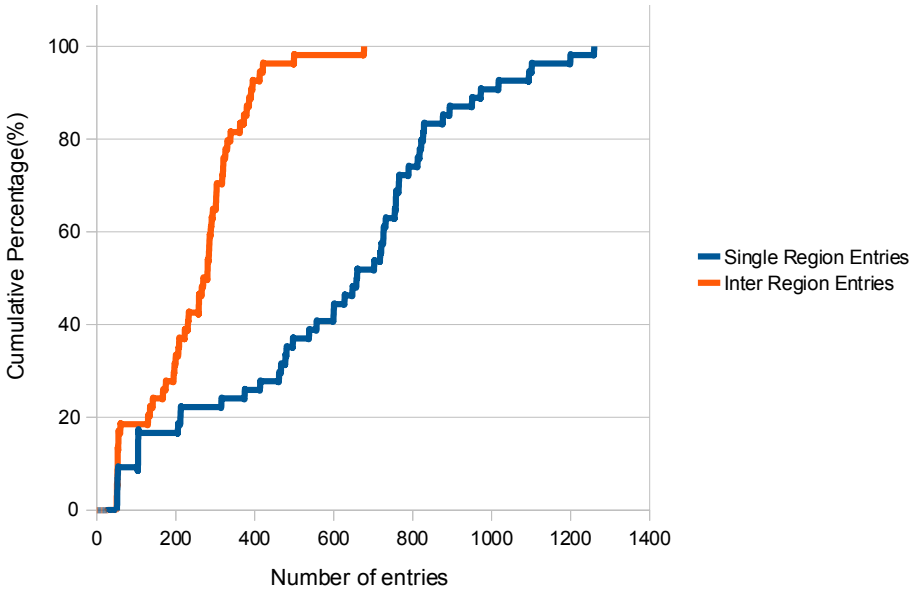

Figure 7.7: Comparison of the number of routing entries for the single region and inter region cases

7.3 Traffic management

To test the TE protocol we need to perform packet level simulation and so we implemented both the pure and the selective feedback versions of the DTIA TE protocol on the *ns2* simulator [ns2]. The protocol uses the DTIA algorithm as the base to infer the remote paths and therefore it was implemented as an extension of the DTIA routing algorithm. Implementation details can be found in [Sil10]. As opposed to the DTIA routing, the TE protocol acts on a subset of the network affecting only the ASes forwarding traffic through the congested links. Therefore scale is not a primary issue. We used the topology from Figure 7.3 for the experiments.

In the current Internet a small set of ASes are the sources of a large percentage of the traffic. These ASes tend to be placed near the *tier-1* transit ASes [LIJM+10]. Traffic consumers are usually part of stub ASes that are less connected. Taking this in consideration we define sets of possible traffic senders and receivers from which we randomly choose three traffic sources and one destination. Initially all ASes distribute an equal amount of traffic through all paths of the highest preference set. The traffic sources are set so that the traffic capacity of links is exceeded. We experimented with and without the TE protocol running and used 110 different sets of traffic origins and destinations to

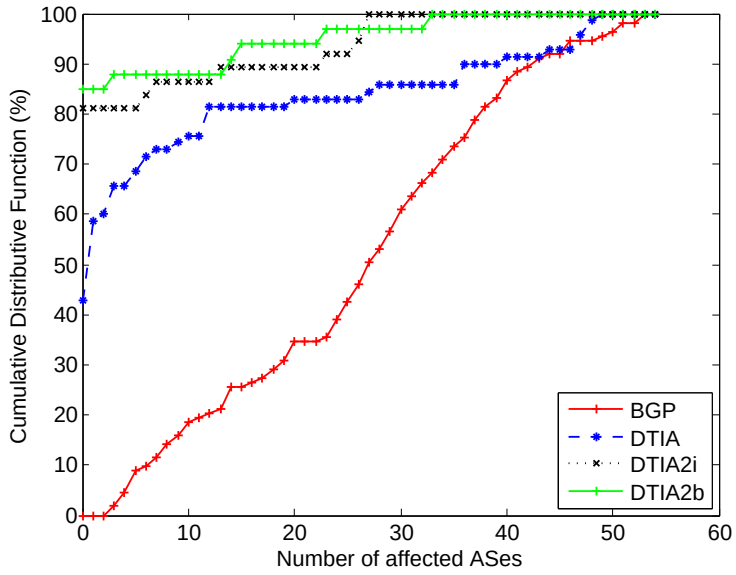

Figure 7.8: Cumulative distributive function (%) of affected ASes after a single link failure

reduce the error margin. We measured the amount of packets dropped in every congested link m_i. Figure 7.9 shows the cumulative distribution function (CDF) of the reduction in relation to the non-TE case (in %) of the number of drops for the 110 experiments.

It shows the results for both feedback versions. We can see that in 50% of the cases (0,5 in the simulations CDF) we have a reduction of at least 90% in the number of dropped packets in the network when the TE protocol is acting. The results are very similar for the two feedback mechanisms with the selective feedback getting marginally better performance. Just as a comparison if we consider a threshold value of 50% of reduction in the number of dropped packets we can see that for the pure feedback case we had 70% of simulations where the reduction is higher while in the selective feedback 80% of the simulations had a higher reduction.

Another important aspect to measure was the traffic distribution before and after the use of the TE protocol. We started by measuring the distribution of traffic amongst the links of the set M of incoming links in the AS detecting the congestion AS_{IN} (including the congested link). To measure the distribution evenness we used the Jain coefficient given by:

$$\frac{(\Sigma_{i=1}^{n} m_i)^2}{(n\Sigma_{i=1}^{n} m_i^2)}$$

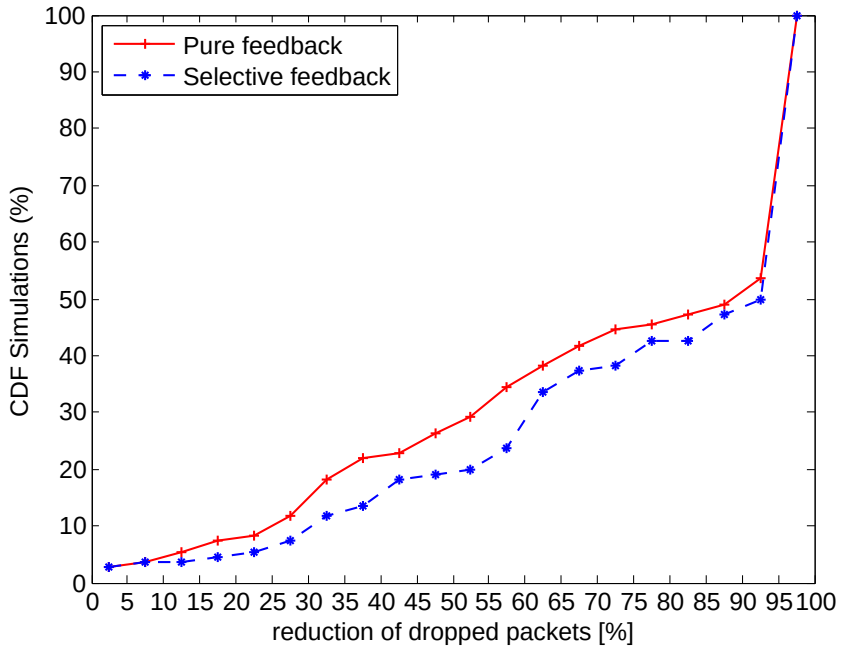

Figure 7.9: Reduction of dropped packets in relation to the non TE case (in %)

where n is the total number of links in M. The coefficient has the value 1 when all links receive the same amount of traffic. Figure 7.10 shows the CDF of the Jain coefficient for all congested link occurrences, measured for the 110 experiments. We can see that for 70% of the experiments the coefficient is below 0.5 without using the TE protocol. Using the TE protocol the percentage dropped to 35%. Just as for the case of the dropped packets we can also see here very similar results between the pure feedback and the selective feedback versions. These numbers show that if all ASes cooperate upon the receipt of a CFP packet we obtain significant results in congestion reduction and better traffic distribution, with a very low cost in terms of complexity on top of DTIA.

This shows that DTIA can form a good base for TE protocols. In our case the major complexity added is the CFP packet distribution. We measured the total number of CFP packets distributed for both the selective and pure feedback versions. In the pure feedback case we had a total number of 2551 CFP packets in the 110 experiments. That number decreased to a total of 1012 packets for the selective feedback case. This reduction of 61.4% is the most significant difference between the two versions. The selective feedback version is therefore an optimization to reduce the number of packets sent and achieves

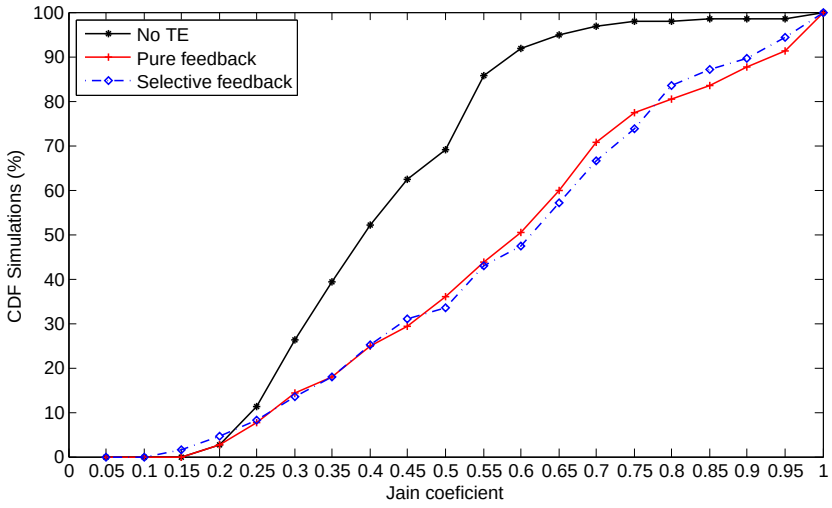

Figure 7.10: Jain coefficient value

similar or better results in the reduction of congestion and even distribution of traffic. This means that it is the most promising version and illustrates what can be done in TE terms using the routing information available in DTIA.

This chapter concludes the presentation of the DTIA architecture. In the next chapter we provide a more generic discussion on multipath routing protocols based on policy to study the practical implications of meeting the correctness conditions in the operation of such protocols.

Chapter 8

Multipath routing using policies

8.1 Introduction

DTIA is a proposal for routing in the inter-domain scenario based on the business policies usually applied in the Internet. It was built from an analysis of the problems found in the current architecture. Most of the problems have roots in the policy nature of the BGP protocol. Traditional shortest path routing protocols find the best routing paths according to a single metric that is minimized. This has well known convergence conditions that can be formalized using graph theory and algebraic constructs. As we have seen in chapter 4, section 4.4, shortest path protocols can be modelled by semirings which have well understood properties that assure convergence in distributed routing with destination based hop-by-hop forwarding. However, working with policy routing leads to different algebraic structures where some of the semiring properties are lost. The main reason is that a policy is a richer concept than a simple metric. A police defines a generic quality of a path and/or link, and the path preference depends on it in a richer way than in traditional shortest path routing. The example of the values for the link labels of DTIA is illustrative. This thesis concentrates on a few characteristics that are fit for inter-domain routing scenarios and that are more difficult to model. These characteristics are:

- Policy routing. Path preference depends on a richly semantic policy rather than on a numeric metric. This means that the total path value is not a simple sum of link metrics but instead a result of the semantic meaning of each link. Another

consequence is the existence of combinations of links that are dismissed (meaning that some paths are invalid).

- Multipath routing. More than one path can be used for forwarding simultaneously. Paths used simultaneously must have the same preference to assure convergence.

- Destination based hop by hop forwarding. The routing solution must be consistent amongst the nodes so that forwarding loops do not occur.

In chapter 4 the literature on algebraic models is reviewed covering the recent work on modelling routing protocols that cannot be modelled by semirings. This work started by trying to model BGP and the current policies used in the Internet. More recent work focused in recasting these models to a purely algebraic setting. In chapter 4 these concepts are systematized and the key algebraic properties that assure convergence in protocols with the above characteristics are isolated. We then used these results to prove DTIA's correctness, but a more generic analysis on protocols with such characteristics is quite interesting. We set a clear objective: a generic policy routing protocol featuring multipath. Then, using the algebraic analysis we assess the trade-offs involved in building such protocols. We start by having no restrictions other than the ones that the resulting algebraic model must fulfil, that being at least one of the convergence conditions stated in chapter 4, section 4.4.4.

8.2 Generic multipath policy routing protocol

8.2.1 Protocol algebraic model

If we look to the way networks are organized we can identify hierarchies. There is a first separation between access/border areas formed by routers providing connectivity to hosts or to routers in other networks, and transport/backbone areas formed by routers that interconnect the border/access routers to transport traffic. Then, the transport/backbone can be itself formed by some hierarchical levels. The hierarchy can be based on node degree (i.e. number of links connecting a node), geographic location, economic relationships or any other differentiating characteristic.

Therefore, it is only natural that our policies are based on the concept of hierarchy. We can identify two purposes to use them: i) to define certain routing objectives, allowing the paths to be chosen by a certain attribute that might have no relation to any measurable metric. It can be thought of as a qualitative identification; and ii) to control which paths are known and can be used. By using policies, several paths can end up having equivalent weights even if their number of hops or bandwidth capacity is different.

As our protocol features multipath it should be modelled using a minset algebra defined over an ordered semigroup. We have seen in chapter 4 subsection 4.4.2 that extending the ordered semigroup model using the minset operator to express the use of multiple paths has no implications in the needed algebraic properties for the protocol to converge. Therefore, for simplicity we model our protocol using an ordered semigroup and analyse its correctness. The minset operator can then be used to express the use of multiple paths simultaneously.

The network is modelled by a directed graph $G(V, E)$ where V is the set of vertices and E is the set of edges. Policies are expressed by values $l \in S$, and each edge $e \in E$ is labelled by a value l. The path weight is obtained by a path composition operation $\otimes$ that defines the weight $w \in S$ for the path. The paths are then ranked according to the preference order $\preceq$. S also contains the elements $\overline{0}$ and $\overline{1}$ representing respectively the weights of the non-existent or invalid path, and the weight of the trivial path.

As we are interested in policy routing, the elements of S need not be numeric - they represent the policy values for links and paths. The way values are assigned to links creates a specific path distribution for each network graph with consequences to having more or less path diversity. Also, the definition of the set of values in S and the way they are composed with $\otimes$ define the possible types of paths and how traffic can be distributed in the network.

Hierarchies are two-dimensional structures and so three elements in S are enough to express all the policies. The three values represent the following policies:

1. D_W Connections in the downwards direction of the hierarchy.

2. U_W Connections in the upwards direction.

A link between two routers which are not at the same level of the hierarchy is

Table 8.1: $\otimes$ operation

$\otimes$	$\overline{1}$	D_W	S_L	U_W
$\overline{1}$	$\overline{1}$	D_W	S_L	U_W
D_W	D_W	D_W	$\overline{0}$	$\overline{0}$
S_L	S_L	S_L	S_L	$\overline{0}$
U_W	U_W	U_W	U_W	U_W

modelled by two directed edges in the graph: one in the upward direction (U_W) and another in the downward direction (D_W).

3. S_L Links between routers at the same level of the hierarchy (peer routers).

Trivial paths ($\overline{1}$) correspond to the highest preference possible; invalid paths ($\overline{0}$) have the lowest preference. So, we have $S = \{\overline{1}, D_W, S_L, U_W, \overline{0}\}$. Path preference is given by the total order $\prec$ and all paths with the same policy value have equal preference. We set the path preference so that D_W paths have the highest preference followed by peer S_L paths and finally U_W paths:

$$\overline{1} \prec D_W \prec S_L \prec U_W \prec \overline{0} \tag{8.1}$$

The $\otimes$ operation is defined in Table 8.1.

There are some observations to make here.

First, the elements of S and the relation of order can be interpreted in abstract terms without any meaning to what a "lower hierarchical level" is in a concrete network or how a node should be considered of higher level than another. However, they define how paths are calculated and how traffic can flow in the network, and the hierarchy relationships between the nodes should be chosen accordingly to the desired behaviour.

Note that the weight of a path is determined by the value of the lowest preference link in the path. This allows for a great number of paths to have the same value and be used simultaneously. It also provides the following features: a path whose weight is D_W only contains D_W links; S_L paths are formed by links with labels $l \in \{S_L, D_W\}$ with at least one S_L link; and U_W paths can be formed by links of any type having at least one U_W link. Paths going in the downward direction of the hierarchy are more preferred than the ones that go through at least one link between nodes at the same level. The least

preferred paths are the ones that contain at least one link to a higher level.

Looking at Table 8.1 we see that there are three cases where an invalid entry was placed. The reason is related to lack of strict monotony in the algebra and the number of possible *non free* cycles that cannot be present in the network.

We use the definition 4.12 described in subsection 4.4.3 to analyse the possible *non free* cycles. We start by defining the sets L_s for each $s \in S \setminus \overline{0}$. Without invalid paths we have the following sets:

- $L_{\overline{1}} = \{\}$ because there is no element $l \in S$ such that $\overline{1} = l \otimes \overline{1}$.
- $L_{D_W} = \{D_W\}$ because only for $l = D_W$ we have $D_W = l \otimes D_W$
- $L_{S_L} = \{D_W, S_L\}$. Since $S_L = D_W \otimes S_L$ and $S_L = S_L \otimes S_L$
- $L_{U_W} = \{D_W, S_L, U_W\}$. Since $U_W = D_W \otimes U_W$, $U_W = S_L \otimes U_W$ and $U_W = U_W \otimes U_W$

Remember that a cycle is *non free* if all its links belong to one of the sets L_s (at least one). Consider L_{U_W}. It contains all possible link values. Then, all cycles are *non free*. This means that the network would have to be acyclic and therefore without multipath. As an example consider two (or more) disjoint paths from a source at a higher level to a middle node at a lower layer and from there to a destination. All of the disjoint parts could not exist simultaneously because they imply a cycle in the network in which all links belong to L_{U_W}.

To solve this we need to change the algebra such that some of the L_s sets have less elements. By invalidating the elements showed in Table 8.1 we erase one possibility on the third item above and two possibilities on the fourth (and last) item above. The new L_s sets are the following:

- $L_{\overline{1}} = \{\}$, remains the same.
- $L_{D_W} = \{D_W\}$ because only for $l = D_W$ we have $D_W = l \otimes D_W$
- $L_{S_L} = \{S_L\}$. Since we now have $\overline{0} = D_W \otimes S_L$ and only for $l = S_L$ we have $S_L = l \otimes S_L$

- $L_{U_W} = \{U_W\}$. We now have $\overline{0} = D_W \otimes U_W$ and $\overline{0} = S_L \otimes U_W$. Only for $l = U_W$ we have $U_W = l \otimes U_W$

We invalidated three types of paths:

- $\overline{0} = D_W \otimes U_W$. This means that a path towards the destination with the weight U_W (goes up in the hierarchy at least in one of its links) cannot be extended with a downwards link (i.e., coming from upwards to proceed again upwards forming a valley);

- $\overline{0} = S_L \otimes U_W$ means that a path similar to the one in the previous item cannot be extended with a link at the same level (this forces traffic to use only once the same level links and always after having gone up. Once the same level link is used the path cannot go up again); and

- $\overline{0} = D_W \otimes S_L$ means that a path with links at the same level and downwards towards the destination cannot be extended with another link coming from a higher level.

Remember that a cycle is *non free* if and only if all its links belong to either L_{D_W}, L_{U_W} or L_{S_L}, and the situation is improving. The first two situations actually reduce to one: if one traverses in one direction is one case, and if one traverses in the other direction is the other case. But, more importantly, they cannot exist in a well-formed hierarchy. The third situation, a cycle formed by nodes at the same level where all links are S_L is not so straightforward as it may seem. There are some subtle problems. The following two examples are illustrative of these problems, and after presenting them we can define a solution. The first example highlights problems both at path calculation stage and at packet forwarding. Consider the figure 8.1, with links between $a_1, a_2, a_3, \cdots, a_{n-1}, a_n = a_1$ all of type S_L. Consider that the destination is node d and link $L_{a2-d} = D_W$. Then the path weight, being calculated around the cycle (outer arrows in the picture) is maintained at S_L causing the routing solution to not converge in limited time. A link is always appended to a path if it is valid and maintains the preference, S_L. This means that a valid path can be constructed by endlessly circulating around the cycle. At packet forwarding stage forwarding loops can also occur since all a_i nodes in the cycle (except a_2) have S_L paths to d by following any of the directions of the inner arrows of the picture.

For example a_{n-1} has two S_L paths to d via a_{n-2} and $a_1 = a_n$ and each one of these also have an S_L path do d using a_{n-1}. So packets can be trapped in loops between a_n a_{n-1}, a_{n-2}, and a_3. If we look closer, a simple link between two of these nodes can form a cycle with the path being extended back and forth between the nodes without the preference being decreased. In our protocol this only happens for S_L links since the other values have different preferences according to the link direction and the preference always decreases if a path returns to the previous node.

We can consider now a cycle with D_W links to see if the problem persists. Consider that destination d is at a lower level than a_2, and a_2 is at a lower level than a_1 and a_3. All other nodes in the cycle are at the same level, S_L. The path calculation stage is now bounded. Consider the following: when the path is being built in the clockwise direction of the cycle, a_3 finds a D_W path and then all other nodes until a_2 have an S_L path. Since $D_W \otimes S_L = \overline{0}$, the path stops in a_1. So, the operation of path calculation stage is no longer infinite. What happens in terms of packet forwarding? Note that there is a sequence of nodes at the same level that find S_L paths to d both in the clockwise and anticlockwise directions.

Therefore, possible forwarding loops can happen. In more generic terms forwarding loops can happen every time there are two or more consecutive S_L links. In these circumstances there is at least one destination node for which two or more nodes have two different S_L paths to reach it. The S_L links have such characteristics that two or more consecutive links of this type in a set of links forming a cycle in the graph forms a non free cycle causing problems in forwarding. This happens because differently from the other two label types these links have the same value in both directions and consecutive S_L links maintain preference. This causes a symmetry in paths with consecutive S_L links in cycles.

A solution for this problem must be found apart from preventing the cycles to exist not to kill the multipath characteristic.

The solution we adopted was to change the algebra so that adding an S_L link has consequences in terms of preference depending on the direction we travel. We created a sense of direction in S_L links. With this paths traversing nodes at the same level have different preferences according to the different directions. This solution solves both the

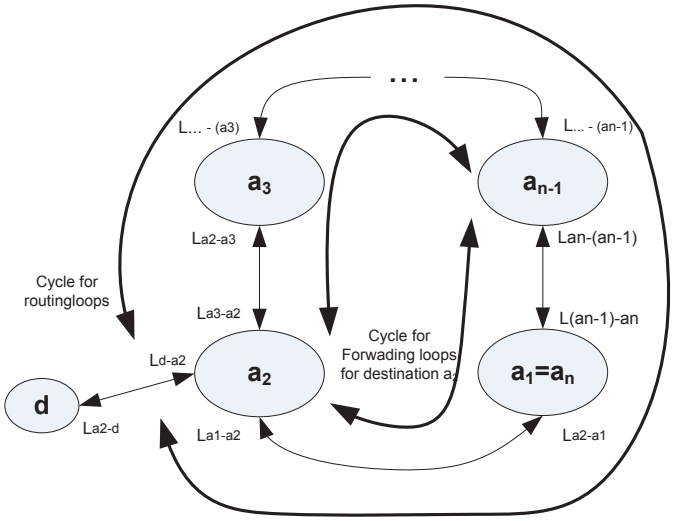

Figure 8.1: Cycle freeness

path calculation problem and the forwarding loops since paths do not have the same preference in both directions of the cycle any more. In practical terms we add a new attribute to the S_L links that denote a secondary preference. Thus, it is modelled by a secondary algebra.

Generically, two different routing metrics can be modelled in an algebraic form by taking the lexicographic product of the two algebras (one for each metric) [BT10b] [GM08] [GG07].

The primary algebra is based on what has been described, and just a small modification on the S_L links is needed. Its ordered semigroup is denoted by $(S, \preceq_S, \otimes_S)$.

The secondary algebra only acts on S_L links. They have an additional secondary label to provide a sense of direction. The secondary label is L (left) in one direction and R (right) in the other. The other link types have an empty value, $\emptyset$, for this secondary label.

The secondary ordered semigroup is $(T, \preceq_T, \otimes_T)$ with the set $T = \{\overline{0}, SI_L, SI_R, L, R, \emptyset\}$. The SI_L and SI_R are only used for path weights, and never as secondary link labels. They are needed to distinguish paths with a single S_L link segment from paths with consecutive S_L links. The $\otimes_T$ operation is shown in Table 8.2:

Table 8.2: $\otimes_T$ operation

$\otimes_T$	$\emptyset$	SI_L	SI_R	L	R
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	L	R
L	SI_L	L	L	L	L
R	SI_R	$\overline{0}$	R	$\overline{0}$	R

Table 8.3: $\otimes_S$ operation

$\otimes_S$	$\overline{1}$	D_W	S_{L1}	S_{L2}	U_W
$\overline{1}$	$\overline{1}$	D_W	S_{L1}	S_{L2}	U_W
D_W	D_W	D_W	$\overline{0}$	$\overline{0}$	$\overline{0}$
S_{L1}	S_{L1}	S_{L1}	$\overline{0}$	$\overline{0}$	$\overline{0}$
S_{L2}	S_{L2}	S_{L2}	$\overline{0}$	S_{L2}	$\overline{0}$
U_W	U_W	U_W	U_W	U_W	U_W

The preference relation on the secondary label is given by:

$$\emptyset \simeq_T SI_L \simeq_T SI_R \prec_T R \prec_T L \prec_T \overline{0}$$

The idea is that amongst paths with consecutive S_L links the preference depends on the secondary label, L or R, of the link with R being more preferred than L. We can now use paths with consecutive S_L links because the sense of direction reduces the number of *non-free* cycles.

Providing navigation at the same level of the hierarchy causes paths through higher levels to be less used. It can be acceptable, but having some control on how S_L links can be used can allow some degree of engineering on the amount of same level routes available. We subdivided the S_L links into two groups: one that limits the number of S_L links on a path to one; and another that allows paths to have multiple S_L links. The primary algebra distinguishes these two S_L link types with two different labels: S_{L1} for the first group, and S_{L2} for the second group (which have then a secondary label, L and R). The set S is now $S = \{\overline{0}, U_W, S_{L1}, S_{L2}, D_W, \overline{1}\}$ and the preference order $\preceq_S$ is now given by:

$$\overline{1} \prec_S D_W \prec_S S_{L1} \simeq_S S_{L2} \prec_S U_W \prec_S \overline{0}$$

The $\otimes_S$ operation is showed in table 8.3.

The total algebra for the protocol is given by the lexicographic product:

$$S \overrightarrow{\times} T = (S \times T, \preceq, \otimes)$$

where

$$(s_1, t_1) \preceq (s_2, t_2) \Longleftrightarrow s_1 \prec_S s_2 \vee (s_1 \simeq_S s_2 \wedge (t_1 \preceq_T t_2))$$

and

$$(s_1, t_1) \otimes (s_2, t_2) = (s_1 \otimes_S s_2, t_1 \otimes_T t_2)$$

This means that paths have the value $\emptyset$ in T until they pass through one S_{L2} link. Then they might have a L or R value in T. If the path has no consecutive S_{L2} links then preference is only dependent on the S set label. When paths have consecutive S_{L2} links the value in T affects preference between equally preferred paths in S. This solves the problems of equal preferences being maintained over consecutive S_{L2} links. A cycle with consecutive S_{L2} links in S is no longer a *non free* cycle, its *freeness* depends on the values of the links in T. A *non free* cycle occurs now if all links in the cycle belong to sets L_1 or L_2 with $L_1, L_2 \subset (S \overrightarrow{\times} T)$ and $L_1 = \{(S_{L2}, R)\}$; $L_2 = \{(S_{L2}, L)\}$. Labelling a cycle in such way makes no sense since it would mean that a node could be reached by a node to its left via a right link. To prevent such cycles a simple rule must be followed.

The rule is that a left-right plane has to be defined for such cycles, leaving at least one node in the opposite side of the others. The links must then be labelled consistently with the plane's division. Take for example the cycle in figure 8.2. All links are labelled S_{L2} in S. In order for the cycle to be *free* a left right plane must be defined separating nodes to the left from nodes to the right. There must be at least one node in each side of the plane. In the example the plane separates node a_3, to the left from all other nodes in the cycle that are considered to the right. The labelling of the links can be done in two ways. The first is following the anticlockwise direction of the cycle: $L_{a_3-a_2} = L_{a_2-a_1} = L_{a_n-a_{n-1}} = L_{a_{n-1}-\cdots} = L_{\cdots-a_4} = (S_{L2}, R)$ since all these links connect nodes to the right of the plane in the anticlockwise direction and $L_{a_4-a_3} = (S_{L2}, L)$. Note that all links having label R in one direction have label L in the other. The cycle is obviously *free* since $L_{a_4-a_3} \notin L_1$. The second way is considering the clockwise direction. In this case we have:

$L_{a_3-a_4} = L_{a_4-\dots} = L_{\dots-a_{n-1}} = L_{a_{n-1}-a_n} = L_{a_1-a_2} = (S_{L2}, R)$ and $L_{a_2-a_3} = (S_{L2}, L)$. Once again, with such labelling the cycle is *free* since $L_{a_2-a_3} \notin L_2$.

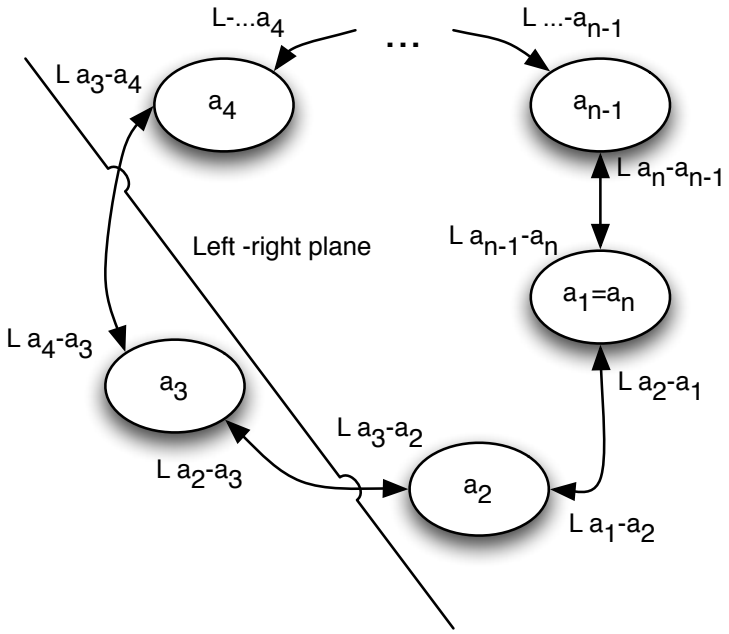

Figure 8.2: S_{L2} cycle right and left separation

The following section proves the correctness of the protocol.

8.2.2 Protocol correctness

To prove that the protocol finds a local left optimal routing solution in finite time, and produces no forwarding loops using destination based hop by hop forwarding, we need to prove that $(S \overrightarrow{\times} T)$ has a *non decreasing* $\otimes$ operation. Then we need to see which *non free* cycles can exist for $(S \overrightarrow{\times} T)$.

We start by proving that the total $(S \overrightarrow{\times} T)$ is *non decreasing*. In [GG07] the lexicographic product is studied, and theorems are derived that relate the algebraic properties of the individual algebras to the properties of their lexicographic product. In our case both S and T algebras are not strictly monotone and are both *non decreasing*. The first step is to check if their product is also non decreasing. According to [GG07] the lexicographic product is *non decreasing* (ND) if :

$$ND(S \overrightarrow{\times} T) \Longleftrightarrow I(S) \vee (ND(S) \wedge ND(T))$$

Where $I(S)$ means that S is *increasing*. So, although S is not stricly *increasing*, both S and T are *non decreasing* and therefore $S \overrightarrow{\times} T$ is also *non decreasing*.

The next step is to see which are the *non free* cycles according to definition 4.12. We have seen that for S the problematic case is for the set L_{S_L}. With the changes we made in S this set no longer exists and now there are two new sets $L_{S_{L1}}$ and $L_{S_{L2}}$:

- $L_{S_{L1}} = \{\emptyset\}$ because there is no value $l \in S$ such that $S_{L1} = l \otimes_S S_{L1}$.
- $L_{S_{L2}} = \{S_{L1}, S_{L2}\}$. Since we have $S_{L2} = S_{L1} \otimes S_{L2}$ and $S_{L2} = S_{L2} \otimes_S S_{L2}$.

So, as long as the S algebra is concerned, a cycle is *non free* if all its links have either the S_{L1} or S_{L2} values. For the total $(S \overrightarrow{\times} T)$ algebra the sets are the following:

- $L_{S_{L2} \overrightarrow{\times} L} = \{(S_{L2}, L)\}$ because only for $l \in S \overrightarrow{\times} T = (S_{L2}, L)$ we have $(S_{L2}, L) = l \otimes (S_{L2}, L)$.
- $L_{S_{L2} \overrightarrow{\times} R} = \{(S_{L2}, R)\}$ because only for $l \in S \overrightarrow{\times} T = (S_{L2}, R)$ we have $(S_{L2}, R) = l \otimes (S_{L2}, R)$.

This means that a cycle is *non free* if it is made of either only (S_{L2}, L) or only (S_{L2}, R) links. Labelling a cycle in such manners makes no sense since it implies that a node that is considered to the left can be reached by consecutively traversing right labelled links. Therefore, the network must not have such cycles as the algebra allows them.

In conclusion the protocol is correct if the network fulfils the following two conditions:

1. No loops exist that do not respect the hierarchy.
2. S_{L2} edges are labelled in T such that it is impossible to form a cycle following only (S_{L2}, L) links or only (S_{L2}, R) links.

This concludes our proof.

8.2.3 Characteristics

We have clearly identified a trade-off between the amount of possible equally preferred paths (that depends on the path composition operation and preference order) and the

network restrictions needed for the protocol to work correctly. In order to assure correctness without imposing too many restrictions on the network topology, some paths had to be considered invalid. Also, although there might be multiple paths we must respect the preference order. This means that we can only use the network path diversity to some extent.

Another aspect is the way traffic behaves given the characteristics of our algebra shaped by the preference order we defined. It is important to understand these characteristics to perform the labelling of the links in the network in order to achieve the best possible use of the network's path diversity. The following characteristics are the most important ones.

Gravity is towards lower level nodes. This means that traffic flows through lower level nodes whenever possible. If a single path via a lower level node is available this single path is used even if multiple paths through same level or higher level nodes exist. All paths via lower level nodes can be used simultaneously.

Paths through same level nodes are only of the same preference if they have the same label in T. So, multiple paths via same level nodes can only be used simultaneously if they are in the same direction (all starting with a (S_{L2}, R) link or all starting with (S_{L2}, L) links).

Paths via higher level nodes are used when they are the only option and all paths via higher level nodes can be used simultaneously.

A particular impact on the amount of available paths for a given node happens when same level links are used. These links allow traffic to flow through paths with a smaller hop count to nearby nodes, but invalidates the use of paths through higher nodes, since the equal level nodes are preferred. It might make sense but has some side effects: consider a regional clique of routers at a certain hierarchical level. Consider a router connected to this regional clique via multiple U_W paths (either upwards directly to the clique, or through intermediate nodes even higher in the hierarchy or in a mid level between the router and the clique). It can use all these multiple paths simultaneously. However, if we add one same level link between the router and one of the regional clique routers, then all routers in the clique are reachable via this link. This addition might reduce the hop count

to reach the clique but it certainty decreases the amount of available paths to just one.

Chapter 9

Conclusions

This thesis has two main contributions. The first is an inter-domain routing architecture featuring multipath. Its design was inspired by the current Internet business model and by the intrinsic characteristics of such a network. It tackles some of the current open issues in inter-domain routing namely the lack of correctness guarantees, the lack of multipath abilities and the poor TE capabilities. We named it DTIA (Dynamic Topological Information Architecture) . The second contribution was derived from the DTIA architecture and concerns multipath routing protocols that are based in semantically rich policies. Obtaining correctness conditions for such protocols has been the subject of recent work in algebraic routing models. In this thesis we isolated the main results from the existent work and provided a simple set of possible conditions that need to be met such that these protocols operate correctly. We then used those results and designed a generic multipath policy routing protocol. In this design we studied the limits on providing multipath routing with semantically rich policies maintaining destination based hop by hop forwarding.

Much of the current open issues in the inter-domain routing architecture used in the Internet derive from the extreme flexibility of BGP and the distributed and independent nature of the network. BGP's flexibility allows for almost any type of routing policy to be set in practice. This together with the distributed nature of the network (with each AS being managed by different entities with different goals) results in a routing architecture without any correctness guarantees. Providing traffic engineering in the current network is also an open problem: it is limited to local traffic manipulations that are made by

altering the distribution of routes or changing their attributes and thus increases the overall system behaviour unpredictability. This thesis starts by focusing on the problem in a more systematic point of view: DTIA separates routing from everything else - a multipath routing solution is found and other concerns like traffic engineering are performed by distributing traffic on top of that solution, but without changing it.

DTIA was also designed with the separation of routing protocol into policy and algorithm in mind. This separation was first introduced by the algebraic routing model community as a systematic way to analyse a routing protocol and it is an elegant way to model a routing protocol. It allows the modelling of DTIA like an ordered semigroup, and the algebraic results shown in chapter 4 provided a clear systematic approach to check its correctness. DTIA's policies are based on the "business model" of the Internet and are limited to a set of possible values for links and paths. This reduces the expressiveness of the routing architecture, and therefore the amount of different possible network configurations. But the fact that this set of policies exist and is followed by every node in the network is a necessary condition to assure that the protocol operates correctly providing an overall coherence. The preference order between path values also follows the current business model. DTIA offers a multipath routing base following this business model that is proven to operate correctly being suitable for operation in an Internet-like network with independent destination based hop by hop forwarding decisions. The possibility to have more than one forwarding path to a destination has also advantages in terms of the failure handling mechanism. Failures are contained locally whenever forwarding is still possible and this results in a reduced number of exchanged messages with a smaller amount of affected ASes than in the current BGP based architecture.

Based on the performed experiments, DTIA's time complexity and space requirements seem scalable to very large topologies in the order of the tens of thousand of nodes. To cope with the even larger scale of the Internet the concept of regions was used. Inter region paths were treated like an independent routing protocol with its own algebra and a different set of policy rules. The same algorithm is used to calculate the path weights but a different $\otimes$ operation and $\preceq$ order were defined. Regions provide scaling benefits from reducing the number of nodes (the number of nodes has an impact in the time and

space complexity of DTIA) but they also increase the fault isolation properties of DTIA reducing the number of affected ASes in case of failures.

Two relevant issues in DTIA are the use of a static map of the region and the use of AS numbers as locators for forwarding. Concerning the map, one could ask what could be the motivations for an AS to keep the network map up-to-date including stating the types of its links? For the *p2c* and *c2p* links the reason is obvious – money. If the link is known more traffic passes through it. A *p2p* link might be in the map, or not. The adjacent ASes know the link exists, so there is no need to put it on the map. But the AS should be interested in putting it in the map for its clients to use it instead of routing their traffic over other valid paths through other ASes diverting the traffic away. *p2pbk* is also obvious because financial advantages can be agreed on its usage (both in active and in standby mode). Finally *p2patt* has an obvious advantage of being known when both ASes belong to the same organization. When they do not belong to the same organization, these types of links can be used to form cliques (at regional or city level). New business models can also be defined using this type of links (charging third party packets, for instance). What is interesting to see is that we can find monetary reasons for the ASes to participate in the building of the map, since routing is performed with business metrics.

The use of AS numbers for forwarding introduces a locator-identifier separation with the need of a mapping service. By separating them, simpler and more appropriated solutions can be defined for each one and the Internet can evolve in a simpler way than today. One characteristic of the Internet that is preventing the speed of the introduction of new solutions compared to other architectures such as for instance the cellular systems is the overload of features in a very small set of entities. Any change has tremendous consequences. DTIA breaks with this tradition. For instance, we could think on another TE protocol without having to change the routing. The mapping service between prefixes and ASes could have mobility features – an entity could keep its address and change AS. Incidentally, prefixes have no meaning in our system. What is important is a translation mechanism between an identifier and an AS. Mapping services are common trait for any system that separates locators from identifiers and are subject of research. In this thesis we only briefly discuss what DTIA mapping service should be. This is, however, a challenging

problem whose solution is outside the scope of this thesis.

DTIA's TE protocol added a simple signalling packet and used only local link information and DTIA features to avoid congestion and achieve better traffic distribution on top of DTIA. The results were encouraging with significant gains (compared to regular DTIA) obtained at such small complexity cost. This indicates that DTIA can be a better base for TE with results being possible with a small complexity increase. As future work it would be interesting to see if DTIA's separation of concerns can provide a base for a solvable formal traffic distribution optimization problem, without adding much more coordination between the ASes.

Changing the Internet routing architecture is a very challenging task and the adoption of something like DTIA should find resistance in such a massively deployed structure. In terms of business, DTIA maintains the current model so that the semantic meaning of paths is not very different from the current one. However, it compromises flexibility, and does not allow things like controlling incoming traffic (although this is also quite limited in BGP). The need that DTIA introduces to publicize the business nature of the AS links (to increase monetary gain), might also be a point of resistance. This is commonly considered a secret by the operators although it is a very open secret nowadays. Finally a complete definition of the static map is needed and it can also be another point of resistance for the adoption of DTIA in the Internet.

The development of the DTIA architecture led to some considerations about the main characteristics that make inter-domain a challenging problem. They are: the distributed nature of the network with each node configured by an independent entity; the use of semantically rich policies that use link and path metrics with a richer meaning than simple numeric properties like distance or bandwidth and impose the existence of invalid paths; the algebraic models for protocols with such characteristics that usually do not have the necessary properties for a global optimum routing solution to be achieved. As it was showed in this thesis in order to maintain destination based hop by hop forwarding in such routing protocols we have to settle for a left-local optimum routing solution. Another conclusion from the development of DTIA is that multipath routing is rather useful in such networks since it allows a cleaner separation of concerns and takes advantage of the

increasing network connectivity with positive impacts in fault handling. So, in the final part of the thesis we focused on the root of the problem and left the Internet behind: we focused on the study of generic multipath policy based routing protocols to be used in networks with distributed control and independent nodes.

We designed a routing protocol that ranks paths using hierarchical based policies with no other restrictions. We proved that a multipath routing solution can be found iteratively in finite time, which opens the way for a variety of possible implementation algorithms. We also proved that no forwarding loops occur even with completely independent destination based hop by hop forwarding. This work provides an insight in the fundamental limits and trade-off's involved in achieving such results for these types of protocols.

The usage of hierarchical based policies transforms the path composition operation $\otimes$ in a *non decreasing* operation (i.e merely not strictly increasing). This is something that should be expected for non numeric metrics applied to links. In fact, we believe that using policies to define path weights (based on hierarchy, commercial relationships between nodes, or any other factor) leads easily to merely *non decreasing* $\otimes$ operations. Multipath routing is also another reason for the path composition to be not strictly *increasing* since having the same preference between paths with different values of some metric (e.g. number of hops) increases the amount of simultaneously usable paths.

Having a *non decreasing* $\otimes$ operation imposes some limits. In order to assure finite time convergence to a routing solution the network's cycles must be constrained, meaning that correctness will not be assured for *every* network but only for networks that do not contain *non free* cycles. We also found that without considering any invalid paths any existent cycle in the network is *non free* and therefore the network would have to be acyclic. The figure of an "invalid path" is quite common in policy routing and the reasons to set paths as invalid can be plenty. Invalidating some paths reduces the number of restrictions in the network and in our case only a few cycles stayed problematic. However, invalid paths easily destroy the *monotonicity* of the protocol model meaning that no global routing solution is to be found but only local optimums. This has an impact on how the routing solution is obtained since in order to retain destination based hop by hop forwarding, paths have to be calculated from destination to source.

Finally, the invalidation of paths and the preference order restrict the amount of path diversity in the network that can actually be used for multipath routing. This is an interesting trade-off since if on one hand a *non decreasing* $\otimes$ increases the number of possible equal preference paths but on the other it means that some paths have to be invalidated (possibly reducing path diversity) or no multipath at all is possible (since the network has to be acyclic).

In summary, we have the following choices: either the protocol only has multipath routing between paths with equal number of hops (with an *increasing* $\otimes$) and all paths can be considered valid, opening space for a possible *monotonic* ordered semi group model meaning that a global optimum routing solution can be found for *any* network; or we open the possibility to have paths with different number of hops to be considered of equal preference (*non decreasing* $\otimes$) and in that case some paths have to be invalid (otherwise the network would have to be acyclic) and this means that only a left local solution is possible. Obviously, the choice is not merely dependent on the multipath aspect. Paths can be considered invalid for policy reasons and the semantic meaning of the policies can impose themselves a *non decreasing* $\otimes$ operation as is the case of both the DTIA's business policies and the hierarchical based policies of the generic routing model. In fact, if we wish to have semantically rich policies where the policy applied to a single link influences the entire path weight then we end up with an algebraic model with an *non decreasing* $\otimes$ operation.

The practical implications of implementing a multipath policy based routing protocol go beyond the need to prove that it operates correctly. There are future work directions opened by the study of such protocols. It would be interesting to see how the use of a particular set of policies affects the possible traffic distribution in a particular network according to the labelling of its links. The labelling of the graph defines the usable paths and therefore the base on which TE can be performed. It creates a first level of possible traffic flows. Methods to find the most effective labelling of a graph so that a particular goal for the routing solution is met could be found. Some examples could be providing an optimal TE base (by providing more possible paths through the most efficient parts of the network) or the maximization of the mean number of equally preferred paths per

destination.

Bibliography

[ABP09] P. Amaral, L. Bernardo, and P. Pinto. *DTIA: An architecture for inter-domain routing.* In IEEE ICC '09, pages 1 –6, june 2009.

[AGA+09] P. Amaral, F. Ganhao, C. Assuncao, L. Bernardo, and P. Pinto. *Scalable multi-region routing at inter-domain level.* In IEEE GLOBECOM 2009.

[BBAS03] A. Bremler-Barr, Y. Afek, and S. Schwarz. *Improved BGP convergence via ghost flushing.* In INFOCOM 2003, volume 2, pages 927 – 937, 2003.

[BFF07] O. Bonaventure, C. Filsfils, and P. Francois. *Achieving sub-50 milliseconds recovery upon BGP peering link failures.* IEEE/ACM Transactions on Networking, oct. 2007.

[BGP] BGP++ home page, *http://www.ece.gatech.edu/research/labs/MANIACS/BGP++/.*

[BGT04] Tian Bu, Lixin Gao, and Don Towsley. *On characterizing BGP routing table growth.* Computer Networks, 2004.

[BT10b] JS Baras and G Theodorakopoulos. *Path problems in networks.* Synthesis Lectures on Communication Networks, 2010.

[CAI] Caida AS relationships data research project, http://www.caida.org/home/

[CDZK05] J. Chandrashekar, Z. Duan, Z.-L. Zhang, and J. Krasky. *Limiting path exploration in BGP.* In INFOCOM 2005. Proceedings IEEE, march 2005.

[CGG06] C Chau, R Gibbens, and TG Griffin. *Towards a unified theory of policy-based routing.* Proceedings of IEEE INFOCOM, 2006.

[CGM03] Jorge Cobb, Mohamed Gouda, and Ravi Musunuri. *A stabilizing solution to the stable path problem.* volume 2704 of Lecture Notes in Computer Science, pages 169–183. Springer Berlin / Heidelberg, 2003.

[Cha06] C Chau. *Policy-based routing with non-strict preferences.* Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications, pages 387–398, 2006.

[CL05] R.K.C. Chang and M. Lo. *Inbound traffic engineering for multihomed ass using AS path prepending.* Network, IEEE, 19(2):18 – 25, mar. 2005.

[CR05] M. Caesar and J. Rexford. *BGP routing policies in ISP networks.* IEEE Network, nov.-dec. 2005.

[D.07] FARINACCI D. *Locator/id separation protocol (LISP). Internet-draft, draft-farinacci-lisp-00*, 2007.

[DB08] Benoit Donnet and Olivier Bonaventure. *On BGP communities.* SIGCOMM Comput. Commun. Rev, 2008.

[DdOV07] S. Dasgupta, J.C. de Oliveira, and J.-P. Vasseur. *Path-computation-element-based architecture for interdomain MPLS/GMPLS traffic engineering: Overview and performance.* IEEE Network, july-august 2007.

[DKF+07] Xenofontas Dimitropoulos, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Young Hyun, kc claffy, and George Riley. *AS relationships: inference and validation.* SIGCOMM Computer Communications Review, 2007.

[DLB06] C. De Launois and M. Bagnulo. *The paths toward IPv6 multihoming.* Communications Surveys Tutorials, IEEE, 2006.

[dLQB06] Cedric de Launois, Bruno Quoitin, and Olivier Bonaventure. *Leveraging network performance with IPv6 multihoming and multiple provider-dependent aggregatable prefixes.* Computer Networks, 2006.

[EKD10] Ahmed Elmokashfi, Amund Kvalbein, and Constantine Dovrolis. *On the scalability of BGP: The role of topology growth.* IEEE Journal on Selected Areas in Communications, oct. 2010.

[FMM+04] Anja Feldmann, Olaf Maennel, Z. Morley Mao, Arthur Berger, and Bruce Maggs. *Locating internet routing instabilities.* SIGCOMM Computer Communications Review, August 2004.

[FPMB07] A. Fonte, M. Pedro, E. Monteiro, and F. Boavida. *Analysis of interdomain smart routing and traffic engineering interactions.* In IEEE GLOBECOM '07, nov. 2007.

[Gan09] F. Ganhao. *Multi-region routing.* Master's thesis, Universidade Nova de Lisboa, *http://run.unl.pt/handle/10362/2394, 2009.*

[Gao00] Lixin Gao. *On inferring autonomous system relationships in the Internet.* IEEE/ACM Transactions on Networking , December 2001.

[GG07] A.J.T. Gurney and T.G. Griffin. *Lexicographic products in metarouting.* In ICNP 2007, oct. 2007.

[GLA93] J. J. Garcia-Lunes-Aceves. *Loop-free routing using diffusing computations.* IEEE/ACM Transactions on Networking, February 1993.

[GM08] M Gondran and M Minoux. *Graphs, dioids and semirings: new models and algorithms.* Springer, 2008.

[GP01] T.G. Griffin and B.J. Premore. *An experimental analysis of BGP convergence time.* In International Conference on Network Protocols, nov. 2001.

[GR01] Lixin Gao and Jennifer Rexford. *Stable internet routing without global coordination.* IEEE/ACM Transactions on Networking, 2001.

[Gri08] T. Griffin and A. Gurney. *Increasing bisemigroups and algebraic routing.* Relations and Kleene Algebra in Computer Science, 2008.

[Gri10] T Griffin. *The stratified shortest-paths problem (invited paper).* Communication Systems and Networks, Jan 2010.

[GSW02] Timothy G. Griffin, F. Bruce Shepherd, and Gordon Wilfong. *The stable paths problem and interdomain routing.* IEEE/ACM Transactions on Networking, 2002.

[Gur09] AJT Gurney. *Construction and verification of routing algebras.* PhD thesis, University of Cambridge, 2009.

[GW00] T.G. Griffin and G. Wilfong. *A safe path vector protocol.* In IEEE INFOCOM 2000, 2000.

[HFP+05] M.P. Howarth, P. Flegkas, G. Pavlou, Ning Wang, P. Trimintzios, D. Griffin, J. Griem, M. Boucadair, P. Morand, A. Asgari, and P. Georgatsos. *Provisioning for interdomain quality of service: the MESCAL approach.* IEEE Communications Magazine, 2005.

[HK07] Benjamin Hummel and Sven Kosub. *Acyclic type-of-relationship problems on the internet: an experimental analysis.* In IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, pages 221–226, New York, NY, USA, 2007. ACM.

[HR08] Jiayue He and J. Rexford. *Toward internet-wide multipath routing.* IEEE Network, 2008.

[HRA10] G. Huston, M. Rossi, and G. Armitage. *A technique for reducing BGP update announcements through path exploration damping.* IEEE Journal on Selected Areas in Communications, october 2010.

[IB07] Luigi Iannone and Olivier Bonaventure. *On the cost of caching locator/id mappings.* In CoNEXT '07: Proceedings of the 2007 ACM CoNEXT conference USA, 2007.

[JR05] A.D. Jaggard and V. Ramachandran. *Toward the design of robust interdomain routing protocols.* Network, IEEE, nov. 2005.

[KcFB07] Dmitri Krioukov, k c claffy, Kevin Fall, and Arthur Brady. *On compact routing for the internet.* SIGCOMM Computer Communication Review, July 2007.

[LCR+07] Karthik Lakshminarayanan, Matthew Caesar, Murali Rangan, Tom Anderson, Scott Shenker, and Ion Stoica. *Achieving convergence-free routing using failure-carrying packets.* In SIGCOMM, 2007.

[LGW+07] Jun Li, Michael Guidero, Zhen Wu, Eric Purpus, and Toby Ehrenkranz. *BGP routing dynamics revisited.* SIGCOMM Computer Communications Review, 2007.

[LIJM+10] Craig Labovitz, Scott Iekel-Johnson, Danny McPherson, Jon Oberheide, and Farnam Jahanian. *Internet inter-domain traffic.* In ACM SIGCOMM 2010, 2010.

[LZN04] Sanghwan Lee, Zhi-Li Zhang, and Srihari Nelakuditi. *Exploiting AS hierarchy for scalable route selection in multi-homed stub networks.* In ACM SIGCOMM conference on Internet measurement , 2004.

[Mag07] Kushman; Kandula; Katabi; Maggs. *R-BGP staying connected in a connected world.* In 4th USENIX Symposium on Networked Systems Design and Implementation, 2007.

[MC04] R. Musunuri and J.A. Cobb. *Convergence of interdomain routing.* In IEEE GLOBECOM, 2004.

[MC05] R. Musunuri and J.A. Cobb. *An overview of solutions to avoid persistent BGP divergence.* IEEE Network, nov. 2005.

[MGVK02] Zhuoqing Morley Mao, Ramesh Govindan, George Varghese, and Randy H. Katz. *Route flap damping exacerbates Internet routing convergence.* SIGCOMM Computer Communication Review, August 2002.

[MI08] Laurent Mathy and Luigi Iannone. *LISP-DHT: towards a DHT to map identifiers onto locators.* In CoNEXT '08: Proceedings of the 2008 ACM CoNEXT Conference, 2008.

[MWZ07] D Massey, L Wang, and B Zhang. *A scalable routing system design for future internet.* Workshop on IPv6 and the Future internet, Jan 2007.

[Nor09] I. Norros. *Powernet: Compact routing on internet-like random networks.* In Next Generation Internet Networks, 2009.

[ns2] The network simulator ns-2 (2.33). http://www.isi.edu/nsnam/ns/

[OPW+10] R. Oliveira, Dan Pei, W. Willinger, Beichuan Zhang, and Lixia Zhang. *The (in)completeness of the observed internet AS-level structure.* IEEE/ACM Transactions on Networking, 2010.

[OZZ07] Ricardo V. Oliveira, Beichuan Zhang, and Lixia Zhang. *Observing the evolution of internet AS topology.* In SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications, USA, 2007.

[PAMZ05] Dan Pei, Matt Azuma, Dan Massey, and Lixia Zhang. *BGP-RCN: improving BGP convergence through root cause notification.* Computer Networks, 2005.

[QB05] Bruno Quoitin and Olivier Bonaventure. *A cooperative approach to interdomain traffic engineering.* In Proceedings of EuroNGI, 2005.

[QIdLB07] B. Quoitin, L. Iannone, C. de Launois, and O. Bonaventure. *Evaluating the benefits of the locator/identifier separation.* InProceedings of MobiArch (ACM SIGCOMM Workshop), Kyoto, Japan, August 2007.

[QTUB04] B Quoitin, S Tandel, S Uhlig, and O Bonaventure. *Interdomain traffic engineering with redistribution communities.* Computer Communications, 2004.

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

[RFC07] *Report from the IAB workshop on routing and addressing* http://datatracker.ietf.org/doc/rfc4984/, 2007.

[RIP] RIPE database. http://www.ripe.net/data-tools/db

[RJJR05] Vijay Ramachandran, Aaron D. Jaggard, Aaron D. Jaggard, and Vijay Ramach. *Relating two formal models of path-vector routing.* In Proceedings of IEEE INFOCOM '05, 2005.

[SAM07] Gireesh Shrimali, Aditya Akella, and Almir Mutapcic. *Cooperative interdomain traffic engineering using Nash bargaining and decomposition.* In IEEE INFOCOM, 2007.

[SCE+05] Lakshminarayanan Subramanian, Matthew Caesar, Cheng Tien Ee, Mark Handley, Morley Mao, Scott Shenker, and Ion Stoica. *HLP: a next generation inter-domain routing protocol.* In SIGCOMM, 2005.

[SG10] J Sobrinho and T Griffin. *Routing in equilibrium.* Mathematical Theory of Networks and Systems MTNS, Jan 2010.

[Sil10] E. Silva. *Engenharia de tráfego leve ao nível do inter-domínio.* Master's thesis, Universidade Nova de Lisboa, http://hdl.handle.net/10362/4316, 2010.

[Sob01] J.L. Sobrinho. *Algebra and algorithms for qos path computation and hop-by-hop routing in the internet.* In IEEE INFOCOM, 2001.

[Sob05] João L. Sobrinho. *An algebraic theory of dynamic network routing.* IEEE/ACM Transactions on Networking, 2005.

[SRW11] J. Scudder, A. Retana, and D. Walton. *Advertisement of multiple paths in BGP.* draft-walton-bgp-add-paths-06. txt (work in progress), Jan 2011.

[Uhl04] Steve Uhlig. *A multiple-objectives evolutionary perspective to interdomain traffic engineering.* In Workshop on Nature Inspired Approaches to Networks and Telecommunications, 2004.

[WCC+05] Hui Wang, Rocky K.C. Chang, Dah Ming Chiu, Chang Dah, Ming Chiu, and John C. S. Lui. *Characterizing the performance and stability issues of the AS path prepending method: Taxonomy, measurement study and analysis.* In Proc. ACM SIGCOMM Asia Workshop, 2005.

[XR06] Wen Xu and Jennifer Rexford. *MIRO: multi-path interdomain routing.* In SIGCOMM, 2006.

[Xu08] Dahai Xu. *Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering.* In INFOCOM, 2008.

[YCB07] X.. Yang, D.. Clark, and A.W. Berger. *NIRA: A new inter-domain routing architecture.* IEEE/ACM Transactions on Networking, 2007.

[YMBB05] M. Yannuzzi, X. Masip-Bruin, and O. Bonaventure. *Open issues in interdomain routing: a survey.* IEEE Network, nov. 2005.

[YXW+05] Y.R. Yang, Haiyong Xie, Hao Wang, A. Silberschatz, A. Krishnamurthy, Yanbin Liu, and Li Erran Li. *On route selection for interdomain traffic engineering.* IEEE Network, 2005.

Anexes

Appendix A

DTIA implementation Algorithm

This appendix contains a brief description of the algorithm used in the implementation of DTIA for the experiments described in chapter 7. The full description can be found in [Gan09]. The algorithm is the same for both the JAVA emulator and the ns-2 implementations.

A.1 DTIA Implementation Algorithm

The implementation algorithm performs a depth-first search on the graph. The search is optimized with heuristics to minimize the number of times a vertex is visited. Consider an AS X that wants to perform the exploration. The exploration is based on walks through the graph of the network and all walks begin at AS X. Each walk is expressed by a path data structure identified as P defined as a sequence of explored ASes that do not loop (*i.e.* do not pass through the same AS twice). Each path data structure P was implemented as a list of links belonging to a given walk. Each link of the walk is identified by its index j where $1 \leq j \leq n$ being n the walk length. Several attributes are recorded in the path structure at each index of the walks: the link's label l_j, the AS number X_j where link j terminates and the weight of the path from ASX to AS X_j denoted S_j. The AS number is used to avoid loops. Each path structure has also a tuple t_j with information about the first hop of the walk. t_j contains two elements: FH and d. FH is the AS identifier of the first hop of the walk reaching X_j at index j of P, and d is the distance from the first hop FH until that point. These tuples will be used at each AS traversed by a walk to provide

the information about all possible first hops that reach the AS. It will serve as base for building the final forwarding table.

Table A.1 illustrates the data structure of a generic path structure P. This structure expresses a walk with n links and link j arrives at AS X_j. t_j has the first hop tuple containing the first hop of the walk, FH, and the distance, d_i, to AS X_i. Finally S_j is the weight of the path from the source to AS X_j.

Path P				
j	link's label l_j	X_j	tuple t_j	S_j
1	l_1	X_1	$t_1 = \{FH, d_1\}$	S_1
2	l_2	X_2	$t_2 = \{FH, d_2\}$	S_2
...	...	...	...	...
n	l_n	X_n	$t_n = \{FH, d_n\}$	S_n

Table A.1: Path P's general structure.

As an example, let us observe figure A.1. We have two walks expressed by path structures P_1 and P_2 that connect X to X_2. Applying the DTIA $\otimes$ operation to calculate the signatures, the final result of each path's structure is shown in tables A.2 and A.3.

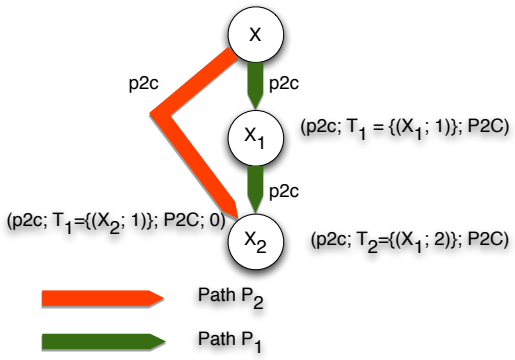

Figure A.1: Figure exemplifying the data structures at each paths' index.

Path P_1				
j	link label l_j	X_j	tuple t_j	S_j
1	$p2c$	X_1	$t_1 = \{(X_1, 1)\}$	$p2c$
2	$p2c$	X_2	$t_2 = \{(X_1, 2)\}$	$p2c$

Table A.2: Path P_1's structure.

To save router resources, we should prevent a walk P that reaches AS X_j under the same conditions as a previous walk Q, from re-exploring the same explored links of walk

Path $P2$				
j	link label l_j	X_j	tuple t_j	S_j
1	$p2c$	X_2	$t_1 = \{(X_2; 1)\}$	$p2c$

Table A.3: Path P_2's structure.

Q after AS X_j. P's segment after X_j will be equal to Q's segment if P reaches X_j under the same conditions, which means: with the same link label l and the same path weight at that point, S. In this case the first hop tuple of walk P should be copied to Q so that all ASes explored by walk Q from the merging point can be marked as reachable via the first hop of walk P. To allow this the path structure is extended and instead of having a single tuple t_j, we have a set of tuples, T_j. Each set contains tuples of diferent walks reaching AS X_i in the same conditions, and from that point onwards those walks are all expressed by the same path structure. Table A.4 illustrates the path structure extended with the tuple set. Each element t_{ji} contains the first hop tuple of walk i at AS X_j.

Path P				
j	link's label l_j	X_j	Set of tuples T_j	S_j
1	l_1	X_1	$T_1 = \{t_{11}, t_{12}, ..., t_{1m}\}$	S_1
2	l_2	X_2	$T_2 = \{t_{21}, t_{22}, ..., t_{2m}\}$	S_2
...	...	...	...	...
n	l_n	X_n	$T_n = \{t_{n1}, t_{n2}, ..., t_{nm}\}$	S_n

Table A.4: Path P's general structure with merging and forking of paths.

Resources are also saved when a path structure forks from a previously explored walk. Let us assume that AS X_j is reached through walk P. Any valid link l_{X_j} from AS X_j with $l_{X_j} \leq l_{max}$, could extend walk P, with l_{max} representing the number of links that extend the walk at X_j forming paths with weights different from $\overline{0}$. One of the links is used by P and the remaining $l_{max} - 1$ links *fork* from P to create new walks that re-use P's explored links. These new walks are expressed in path structures, the first hop tuple set of P is copied to these new structures so that the information of the first hops that reach the forking point is not lost.

A new problem arises at this point: let's consider a walk R that merges into a walk P at index j (AS X_j) and that walk U is forked from P in a subsequent link x. The first hop tuples from R have to be copied to both P and U or any other walk that has forked from P. For this purpose the path structures store pointers to all forked path structures

that depart from it.

For each AS of the topology a list of first hop tuples similar to the one in a path structure is created. The objective is to have for each AS a complete list of first hop tuples for all walks that reach this AS. The process is the following: each time a walk P reaches an AS X_j, its first hop tuple list is compared with the one in the AS. The first hop tuples that are not present in the X_j list are added. The repeated ones are only added if P's weight is better than the previous ones. This information will be used to build the forwarding table. Table A.5 exemplifies the list for AS X_j.

Best First Hops FH_i that reach X_j		
Weight S_i	FH_i	d_i
S_1	FH_1	d_1
S_2	FH_2	d_2
...	...	...
S_{Fmax}	FH_{Fmax}	d_{Fmax}

Table A.5: Best first hop tuples that reach AS X_j

FH_i is the first hop identifier with $1 < i < Fmax$, where $Fmax$ is the number of first hops that reach AS X_i, d_i is the traversed distance to reach X_j via the first hop i and S_i is the weight of the path starting with that first hop.

Having covered the general data structures, we can now explain the general algorithm through the visual aid of flowcharts. The path exploration procedure starts by creating an initial path structure for each of the active links of the source AS X; if X has an active link to each neighbour X_i with $1 \leq i \leq N$, then we can define N single path structures P_w with $1 \leq w \leq N$.

When the path exploration starts at the source, each neighbour AS can be reached by paths with weight $S = l \otimes \overline{1}$, where l is the initial link label. At each initial link with index $j = 1$, we fill the respective values of the path structure table. The values are filled as presented on table A.1 and exemplified on tables A.2 and A.3.

After the initialization of the first element of each path structure P_w, the general algorithm for path exploration begins. Figure A.2 presents a flowchart with the general algorithm. As observed at the flowchart, the general algorithm explores the entire set of path structures by processing each P_w in a hop-by-hop process. A path is processed until it is no longer possible to walk on the network's graph, either because there are no other

links or P_w has merged with another walk. At that point, the algorithm moves to the next walk P_{w+1}. When all walks are processed, the forwarding table is built based on the tuples recorded at each AS.

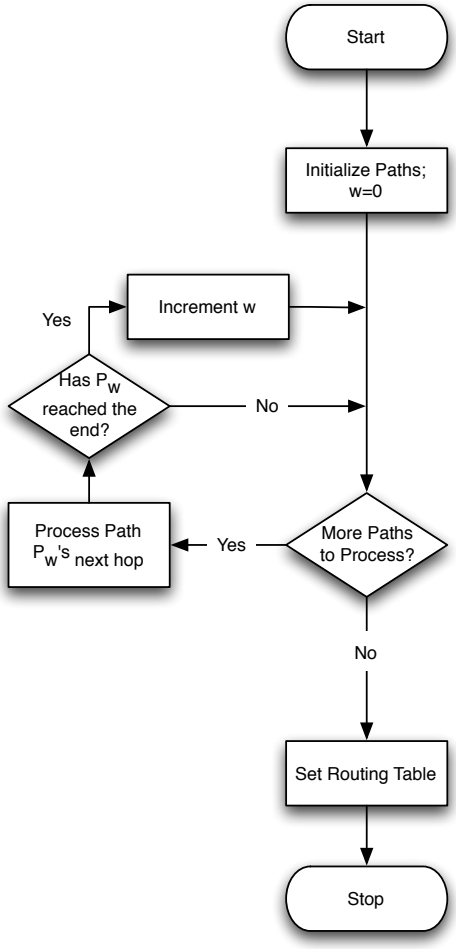

Figure A.2: Path exploration general algorithm.

The flowchart from figure A.3 illustrates the algorithm that explores and validates the walk P_w at each hop. Assume that the last explored AS from walk P_w is AS X_j and that l_i is one of the X_j links, with $0 \leq i \leq l_{max}$ where l_{max} is the number of links of X_j.

From the flowchart of figure A.3 we have two distinct phases for the procedure that processes walk P_w:

1. *Validation* - it defines whether a link is valid or not for further processing by determining the result of the $\otimes$ operation;

2. *Execution* - it *explores* the path P_w with the new link l_i and determines if path P_w

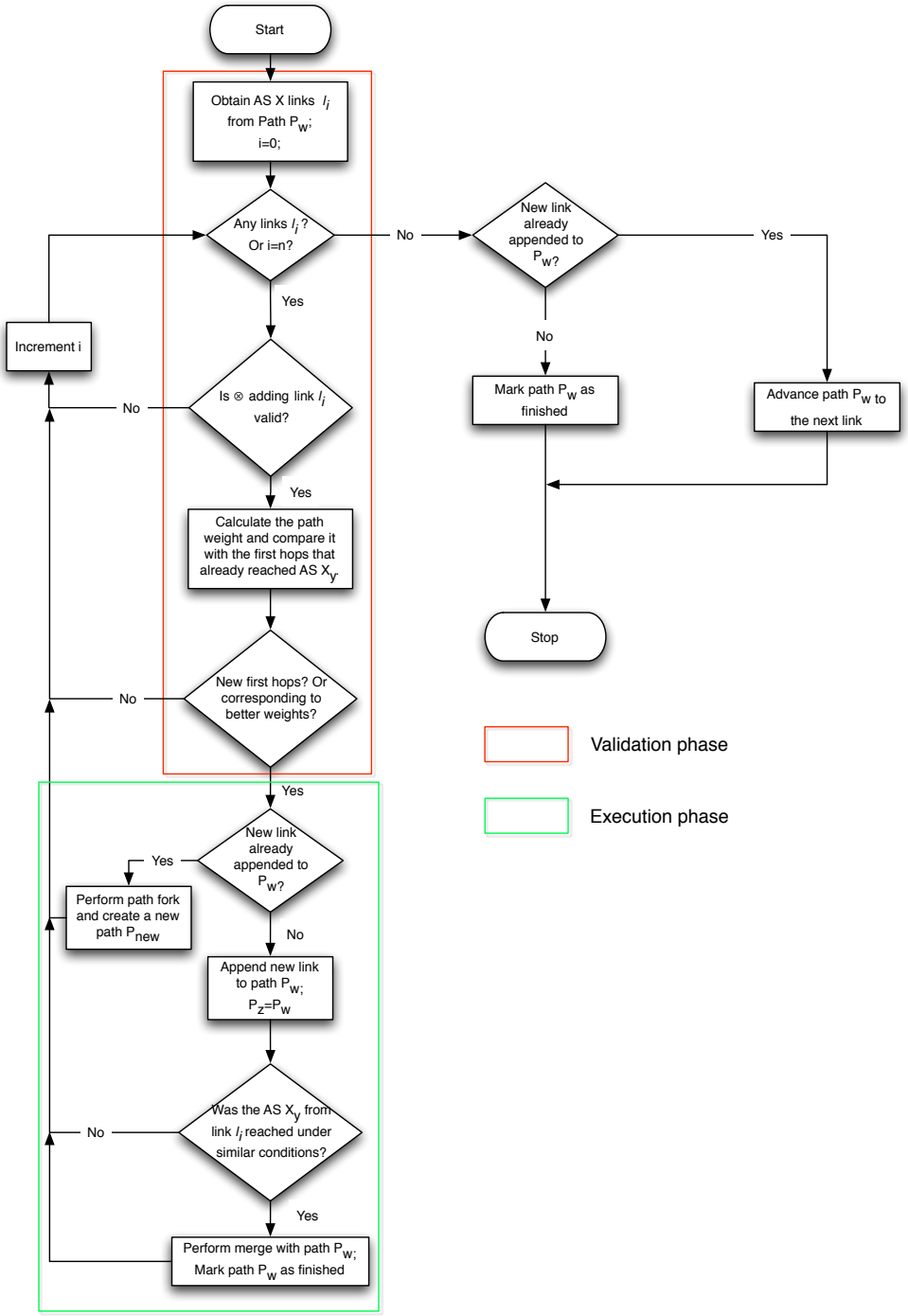

Figure A.3: Flowchart that illustrates the processing of path P_w.

should fork a new path P_{new} or merge into an existing path P_z.

If AS X_j does not have any links l_i, then the algorithm should end by marking path P_w as finished. Otherwise each link l_i from AS X_j is verified in the *validation* phase and explored in the *execution* phase.

At the *validation* phase, it is verified if AS X_y reached by l_i is eligible to extend path P_w, *i.e.* it does not loop and has a result different from $\overline{0}$. Afterwards, the first hops that use this path are compared with the first hops that already reached AS X_y. There are two situations that allow us to proceed to the *execution* phase:

1. Path P_w has new first hops that have not reached AS X_y before;

2. Path P_w has the same first hops that AS X_y has. However, path P_w has first hops with better path weights than AS X_y. This situation allows us to increase the preference of the first hops that reached AS X_y. If the weight S_{P_w} of path P_w has more preference than the weight S_{X_y} of AS X_y, *i.e.* $S_{P_w} \prec S_{X_y}$, then the previous weight should be replaced by P_w's weight.

At the *execution* phase, the algorithm verifies if any link l_i from AS X_j has been appended to P_w. If not, one of such links l_i is appended to path P_w and the exploration of that path structure will continue via that link. For the other links a new path P_{new} is created and forked from P_w. P_{new} *inherits* all of P_w's updated first hop tuples. All valid tuples, *i.e.* new first hops or old ones with better weights, are recorded at the AS reached by the chosen link AS X_y.

For the current link that was appended to P_w, we should verify if it is possible to merge path P_w with another path P_z. To merge both paths, P_w must reach AS X_y under the same conditions as P_z, which means having the same weight S and link label l.

The flowchart from figure A.4 illustrates the merge algorithm when a path P_w merges with a path P_z.

This algorithm is divided in three phases:

1. Copy the set of updated first hop tuples from the previous path P_w to all ASes reached by path P_z, starting at AS X_y;

2. Copy the set of updated first hop tuples to all paths that forked from P_z, starting at AS X_y;

3. If path P_z is merged with a path P_{zz}, the merge algorithm is called recursively to update P_{zz} with the new tuples.

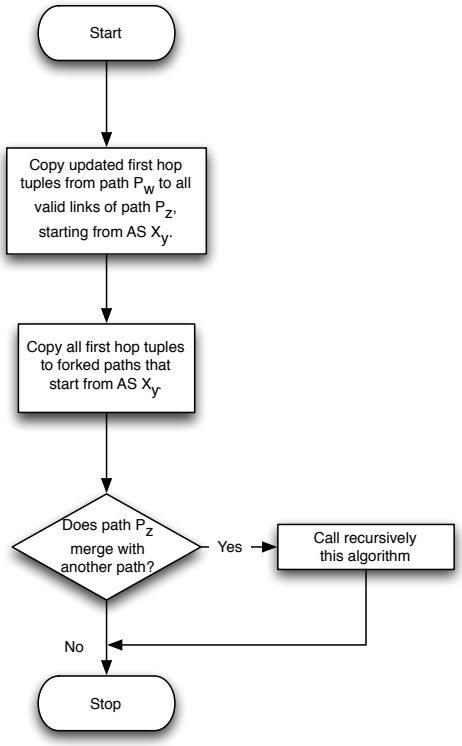

Figure A.4: Merge of a path P_w algorithm.

When all paths are processed, the general algorithm pursues to the construction of the routing table for all destinations. Each destination AS X_j has a list of best first hops that reach it with their respective weight and distance. These first hops are ordered according to the weight and the distance from the source AS X to destination AS X_j. Tuples are ranked according to three rules:

1. Assume that $S_1 \neq S_2$, where S_1 is the weight of the tuple (FH_1, d_1) and S_2 is the weight of the tuple (FH_2, d_2). The first tuple is more preferred than the second if S_1 has a higher preference than S_2;

2. Assume that $S_1 \simeq S_2$, where S_1 is the weight of the tuple (FH_1, d_1) and S_2 is the weight of the tuple (FH_2, d_2). For weights equal to (bkp, x) and $p2patt$, the first tuple is more preferred if d_1 is lower than d_2, *i.e.* FH_1 needs less links than FH_2 to reach the destination. For all other weights both tuples are equally ranked.

3. If $S_1 = S_2$ and $d_1 = d_2$, by default we assume that tuple (FH_1, d_1) is equally ranked as (FH_2, d_2).

To build the multipath routing table, we select for each destination the best ranked first-hops that have equivalent weights.

Best First Hops that reach X_j		
Weight	First Hop	distance
$p2patt$	a	2
$p2patt$	b	9
$p2c$	c	1
$p2c$	d	3
$p2p$	e	3
$p2p$	f	3

Table A.6: Best first hop tuples that arrive destination X_j.

Final Ranking		
Weight	First Hop	distance
$p2c$	c	1
$p2c$	d	3
$p2p$	e	3
$p2p$	f	3
$p2patt$	a	2
$p2p$	b	9

Table A.7: Ranking of the first hop tuples that arrived X_j.

Table A.6 exemplifies the first hop tuples that are recorder at destination AS X_j at the end of the path exploration algorithm. The first hops are enumerated from a to f. Tuples are ranked according to the aforementioned rules to produce the forwarding table. The tuple's ranking is shown on table A.7. In this case the AS can use simultaneously the two first paths with weight $p2c$ to forward packets