João Pedro Gonçalves Chalaça

Licenciado em Ciências da Engenharia Electrotécnica e de Computadores

Multipath Policy Routing in Packet Switched Networks

Dissertação apresentada para obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores, pela Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia.

Orientadores : Paulo da Fonseca Pinto, Professor Catedrático, FCT-UNL

Pedro Amaral, Professor Auxiliar, FCT-UNL

Júri:

Presidente: Prof. Doutor Luís Bernardo

Arguentes: Prof. Doutora Anikó Costa

Vogais: Prof. Doutor Pedro Amaral

Prof. Doutor Paulo Pinto

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Janeiro, 2013

Multipath Policy Routing in Packet Switched Networks

Aos meus pais

e irmã

Agradecimentos

Este espaço é dedicado a todos aqueles que me acompanharam durante a elaboração desta dissertação. Para todos eles, o meu mais sincero obrigado!

Primeiro, gostaria de agradecer a toda a minha família, e em especial aos meus pais e irmã pela compreensão, apoio, a muita paciência durante todo este tempo e por me terem dado a oportunidade de atingir este nível académico. À minha namorada, por toda a amizade, amor, dedicação e por o apoio incondicional ao longo da minha formação. A todos meus amigos em especial ao Fernando, Lauro, Baptista e Sérgio por todas os momentos de descontração e diversão.

Agradeço ao Prof. Pedro Amaral pela orientação, apoio constante ao longo da minha dissertação e pela oportunidade fazer este trabalho de investigação. Um especial agradecimento ao Prof. Paulo Pinto, pelo incentivo na busca de conhecimento, sendo um exemplo de competência, garra, determinação e disciplina.

Ao Departamento de Engenharia Electrotécnica da Universidade Nova de Lisboa, indispensável para a minha formação académica.

A todos os colegas da FCT-UNL pelo apoio, convívio e toda a palhaçada ao longo de todo o curso, com particular destaque para P. Almeida, E. Santos, F. Oliveira, M. Mateus, A. Rocha, J. Manuel, T. Teixeira, G. Azevedo, V. Astúcia, B. Branco, B. Duarte, J. Martelo, C. Lucena e V. Chamorrinha.

Muito obrigado a todos.

Sumário

Hoje em dia, o bom funcionamento das grandes redes é de tremenda importância, pelo que os protocolos de encaminhamento se têm tornado cada vez mais extensos e complexos. O encaminhamento baseado em políticas pode ser de grande importância para redes comuns quando o custo de transportar um bit já não é o ponto fulcral.

O problema do melhor caminho é visto como a generalização do problema do caminho mais curto que melhor se adequa ao encaminhamento baseado em políticas. Isto significa que as preferências nos caminhos dependem de características semanticamente ricas, nas quais dois caminhos diferentes podem ser considerados igualmente preferidos. No entanto, os actuais modelos de encaminhamento com base em políticas não conseguem aproveitar totalmente a multiplicidade de ligações para um determinado destino na rede. Assim, o multicaminho pode trazer várias vantagens ao encaminhamento com base em políticas, pelo que desenhar este género de protocolos de encaminhamento parece ser um problema de interesse.

Para modelar problemas de encaminhamento, são usadas estruturas algébricas e teoria de grafos. Os problemas de encaminhamento podem ser resolvidos através de variantes dos métodos clássicos da álgebra linear. O objectivo desta dissertação é então elaborar um protocolo de encaminhamento multicaminho baseado em políticas, usando um protocolo hop-a-hop baseado no destino, com decisões de encaminhamento independentes. As redes podem então ser mais resilientes, fornecer uma melhor distribuição de tráfego e manter um paradigma de encaminhamento simples. A dissertação conclui com as trocas entre a flexibilidade da solução apresentada e a quantidade de múltiplos caminhos que podem ser usados em simultâneo.

Palavras Chave: Encaminhamento Multicaminho, Políticas de Encaminhamento, Problemas de Encaminhamento, Modelos algébricos.

Abstract

Nowadays, the continuous operations of large networks, under multiple ownerships, are of tremendous importance and as a result, routing protocols have gained numerous extensions and accumulated complexity. Policy-based routing can be of significance for common networks when the cost of transporting a bit is no longer the biggest pressure point.
The best path problem is a generalization of the shortest path problem that suits policy based routing. This means that preferences for the paths depend on semantically rich characteristics, in which two different paths may have the same preference. However, current policy-based routing models cannot take full advantage of the multiplicity of connections to a given destination and are single path in nature. Therefore multipath can bring several advantages in policy based routing.
Designing multipath routing protocols based on policies seem to be a problem of interest. To model routing problems, algebraic structures and graph theory are used. Through variants of classical methods of linear algebra routing problems can be solved.
The objective of this dissertation is to devise a multipath policy-based routing protocol using a simple destination-based hop-by-hop protocol with independent forwarding decisions. Networks featuring these characteristics can be more resilient to failures, provide better traffic distribution and maintain a simple forwarding paradigm. The dissertation concludes with the trade-off's between the flexibility of the proposed solution, the amount of multiple paths that can be used simultaneously and the network restrictions that must be applied.

Keywords: Multipath Routing, Policy Routing, Algebraic Models, Routing Problem.

Acronyms

AS Autonomous System

ASes Autonomous Systems

BGP Border Gateway Protocol

CAIDA Cooperative Association for Internet Data Analysis

DTIA Dynamic Topological Information Architecture

DV Distance Vector

e-BGP External Border Gateway Protocol

ECMP Equal-Cost Multi-Path routing

i-BGP Internal Border Gateway Protocol

IGP Interior Gateway Protocol

IGRP Interior Gateway Routing Protocol

IS-IS Intermediate System - Intermediate System

ISP Internet Service Provider

LS Link-State

LSA Link-State Advertisement

LSP Label-switched path

LOCAL_PREF Local Preference BGP attribute

MED Multi Exit Discriminator BGP attribute

MIRO Multi-path Inter-domain ROuting

MPLS Multiprotocol Label Switching

NIRA A New Internet Routing Architecture

NRLS Name-to-Route Lookup Service

OSPF Open Shortest Path First

QoS Quality of Service

RIP Routing Information Protocol

RIPE Réseaux IP Européens

SPP Stable Paths Problem

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer networks are growing at unprecedented rates and routing is more and more a demanding problem for many reasons: networks are larger, diverse, under several ownerships and its continuous operation is of tremendous importance because it penetrates deeply into the world economy therefore affecting all society. To answer to this increasing demand, routing protocols have been developed and have accumulated numerous extensions and complexities.

Routing may be abstracted to be a path finding problem in a graph and the solution to it has captivated the interest of the scientific community. The most common form of the problem is the shortest path problem, which is to find the path with the lowest cost, where cost is often a measurable numerical metric like: hop count, path bandwidth, path reliability, latency and others metrics. Some common solutions for the shortest paths problem are well known and are the source of many of the most used routing protocols.

The Internet introduced some new requirements for routing protocols. When routing is performed between independently managed networks called domains or Autonomous System (AS, in plural ASes), it 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 policies. Unlike a simple numerical metric, these policies reflect a wider set of characteristics with semantically rich concepts, defining the nature of the paths

and their relative preference. Given these characteristics policy routing provides great flexibility in defining route preferences, and it has been the reason for the use of Border Gateway Protocol outside its original scope of inter-domain routing. Routing based on policies can be of interest in several networks when the cost of transporting a bit is no longer the biggest pressure point of modern networks.

A generalization of the shortest path problem that suits the policy based routing is the *best path* problem. The concept of the best path means that preferences of the paths depend on characteristics such as business relationships with other nodes, defining in this way a hierarchical order amongst the paths. However current policy routing models cannot take full advantage of the multiplicity of connections to a given destination and are single path in nature.

With policy routing two paths that are very different according to traditional numeric metrics can have the same policy characteristics and therefore have the same preference and are considered equally good. Multipath can take advantage of this and therefore be a cornerstone in policy routing. Single-path routing protocols have a critical interval after a failure until the algorithm converges to a new solution. On the contrary, in the multi-path case in the event of a failure, equally good alternative paths might still be available reducing the importance of the re-convergence process. Having multiple paths also means that traffic engineering can be achieved by wise distribution of traffic amongst those paths, instead of having to fiddle with network metrics to obtain the desired result via routing state manipulations.

Designing multipath routing protocols based on policies that can operate using simple destination-based hop-by-hop forwarding seem to be a problem of interest. The main goal of this dissertation is to devise a multipath policy based routing protocol using a simple destination-based hop-by-hop protocol with independent forwarding decisions. The approach was to use a foundation of algebraic structures and graph theory to model the problem. The use of such models has been put to practice to model traditional shortest

path protocols as well as Border Gateway Protocol. This dissertation uses the model described in [1] that provides a way to model such protocols and prove the correct behaviour of the presented routing model.

1.1 Objectives and Contributions

The aim of this dissertation is to develop a generic multipath policy routing model based on destination hop-by-hop forwarding.

The main contributions of this dissertation are the definition of a new generic multipath policy-based routing model that is formally proven to work correctly. Also in the design of the protocol some fundamental limits of such protocols become evident. Some of the most important ones are the trade-offs between the flexibility of the model, the amount of multiple paths that can be used simultaneously and the network restrictions that must be applied. An implementation method for the protocol that shows how a matrix iteration method can be applied to calculate the routing solution is also presented. Finally a set of experiments that shows the routing results that can be achieved for two common and different scenarios are preformed.

1.2 Dissertation Structure

This dissertation is structured in five chapters and the contents of each chapter will be briefly described in this section.

Chapter 2 presents the current state of the art on Internet routing protocols and academic solutions, some pros and cons for each routing solution are analysed. It also contains a brief discussion of how algebras appear in the routing context, and how the routing problems can be both expressed through algebraic models and solved by linear algebra with variants of classical methods. From these studies we identify important characteristics for this dissertation proposed solution.

Chapter 3 presents this dissertation's theoretical model for the multipath policy rout-

ing protocol. A set of conditions that need to be met such that these protocols operate correctly is proposed as the main contribution of this dissertation. In the proposed solution we studied the limits of providing multipath routing with semantically rich policies maintaining destination-based hop-by-hop forwarding.

Chapter 4 describes the implemented algorithms and experimented topologies. Also by comparing the various experiment results obtained in terms of diversity of paths, it performs an analysis performance of the developed protocol.

Chapter 5 presents the dissertation's conclusions and discusses of some possible future work.

Chapter 2

Related Work

This chapter introduces the necessary background on internet routing, and presents the fundamental mathematical concepts needed for a better comprehension of the algebraic models used throughout this dissertation. We start by discussing Internet routing.

2.1 Internet and Network Routing Protocols

The Internet is a decentralized arrangement of networks, where each network needs to know how to reach all destinations. To achieve this, a network needs a routing protocol to determine the available paths and choose the ones that should be used to forward packets to a given destination.

This section presents the most well known traditional routing protocols along with their limitations and advantages. Routing protocols are responsible for creating, maintaining and updating routing tables in routers, which react to network failures and additions. That information is then used to forward packets all over the Internet. These routing protocols can be divided in two distinct levels: Intra-domain and Inter-domain [37].

2.1.1 Intra-domain Routing Protocols

Intra-domain protocols are used in networks that are maintained and managed by the same entity, which means that the same behaviour from all nodes is easy to assure and that the same goals are usually shared amongst them. The usual goal in intra-domain

routing is to forward packets as efficiently as possible from the source to the destination (performance-oriented).

There are two major classes of intra-domain routing protocols: Distance Vector (DV) and Link-State (LS).

2.1.1.1 Distance Vector

In Distance Vector (DV) protocols, routing information is exchanged between directly connected neighbours and it requires that each one computes the best path (output link) to each destination by using the exchanged information. The first routing algorithms developed for ARPANET were DV implementations and one of the most popular was the Routing Information Protocol (RIP) [25]; today the most used is the Interior Gateway Routing Protocol (IGRP) [9].

The node maintains a routing table and once it receives a message (called a vector), it adds the neighbour's link cost to the values in the table and, then computes and updates the best path to each destination. If the distance to any destination changes it notifies his neighbours. This distributed route computation using only neighbour's information corresponds to a distributed implementation of the *Bellman-Ford* algorithm [37].

Even though the concept is rather simple and scalable, DV protocols have the known problem of count-to-infinity. This happens because a node knows from which neighbour a path was learned, but it does not know from where that neighbour learned the path [6]. Measures like split horizon with poison reverse and hold down techniques are employed to avoid the count-to-infinity problem. Despite the improvements these protocols still suffer from slow convergence time to a stable routing solution [37].

2.1.1.2 Link-State

The need for faster convergence times leads to a new class of routing protocols, called LS protocols. Two widely used examples are, Open Shortest Path First (OSPF) [27] and In-

termediate System - Intermediate System (IS-IS) [29].The main difference of this protocol class is the maintenance of an overview of the network as a support mechanism for lower convergence times after network changes.

In order to achieve a lower convergence time, each node periodically floods an advertisement of the state of its links to its neighbours. This message is called a Link-State Advertisement (LSA). When a node receives the LSA from its neighbour, stores it and floods the received message to all neighbours, except the one that originated it. LSAs have a sequence number and the message is dropped if a newer LSA is already present. This mechanism allows a faster detection of failures and accelerates the re-convergence to a routing solution when compared with the multiple exchanges of vectors required by DV protocols.

Another difference to DV protocols is that in LS protocols the calculation of the best paths is made by each node using the overview of the network given by the LSAs. This is typically done using the *Dijkstra* algorithm [8].

Despite being faster to converge to a routing solution than DV, it fails to scale well since each node must store the network's topology and compute the shortest path tree each time a new message is received, and every time a topology change occurs (or link failure/activation) the Link-state information is flooded throughout the network to ensure all nodes are updated, thus consuming more resources than a DV Protocol [41].

2.1.1.3 Multiprotocol Label Switching

There is also another kind of technology that is widely used, especially for intra-domain routing, called Multiprotocol Label Switching (MPLS)[32]. It has a mechanism to carry packets along a pre-determined path that is specified in advance (explicit routing), rather than a traditional routing protocol used for destination-based hop-by-hop forwarding.

Each path is built using labels. When a packet enters the path, the ingress router en-

capsulates the packet in a MPLS header, which includes the first label. Intermediate routers only look at the label when forwarding MPLS packets. The labels are short and require only exact matching. These features make finding the correct output link and forwarding a very quick process. The egress router removes the MPLS label and forwards the packet based on its original header. The "multiprotocol" in MPLS appears because, MPLS headers do not belong to the network layer packet or the data link layer frame, meaning MPLS switches can forward both IP packets and non-IP packets [37].

MPLS allows label stacking to concatenate path segments into longer MPLS paths. The explicit routing provides the basic mechanisms for facilitating traffic engineering. A mechanism to set up paths has to be used. One of the options is to use protocols that distribute knowledge of the topology/bandwidth/etc. using routing information from a traditional Link-State routing protocol.

2.1.2 Inter-domain Routing Protocol

Routing in inter-domain scenario is quite similar to the intra-domain scenario. The main difference is that a large collection of independent nodes with distinct objectives from each other are more characteristic in inter-domain scenarios. Unlike Intra-domain routing protocols, that are based in traditional distance-based numeric metrics, Inter-domain routing is policy-oriented. This means that routers forward packets as a function of local policies that reflect relationships between the participating nodes.

The standard protocol for the Internet Inter-domain routing is the BGP [31] currently in its fourth version. The objective of inter-domain routing is the establishment of paths between separate and independently managed networks called Autonomous System (AS). An AS contains a set of internal routers, with a single routing policy (the decisions of which routes to exchange and which routes to prefer) and controlled by a single entity or administration; Autonomous Systems (ASes) can be Internet Service Providers (ISPs), content providers, small companies, etc.

Border Gateway Protocol

A path vector protocol can be seen as an extension of DV routing, it advertises the entire path for each destination instead of announcing a metric per destination. BGP behaves as a path vector routing protocol, since the BGP routing messages carry a list of ASes along the path to a destination. BGP behaves as a path vector routing protocol, since the information exchanged is entire reachable paths for visible destinations. When an AS receives a path it prepends its identification to the path and forwards the path advertisement to valid neighbours. This way, each message carries the ordered list of traversed ASes, preventing routing loops.

For routers to exchange path information, first they need to establish a TCP session. There are two types of sessions: Internal Border Gateway Protocol (i-BGP) and External Border Gateway Protocol (e-BGP). When BGP is used to learn routes between border routers inside the AS, it is referred to as i-BGP and when used to learn routes between ASes, it is called e-BGP.

Routers typically receive multiple paths to the same destination, so a decision process (filtering procedure) chooses which path is the best to forward traffic and only that path is advertised to the neighbouring ASes. The decision process assigns the first valid path as the current best path and evaluates it against the remaining paths. The evaluation in based on a set of rules used to determine the best path. These rules are processed in order of priority and are shown on Table 2.1, by order of priority and indicating who sets the corresponding attribute values: either the neighbour from which the route announcement is received (Neighbour) or the evaluating node itself (Local Router).

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

Table 2.1: BGP rules for tie-break decision process [31]

Rules 1 and 3 are the most interesting because they are good examples of how complex routing policies may be performed with BGP. The LOCAL_PREF attribute set by the AS administrator, is used to control outbound traffic, by assigning a level of preference to the received paths. The MED attribute is exchanged between ASes, to try to influence the decision of the neighbouring nodes and therefore trying to control inbound traffic. However it is not assured, since the use of the LOCAL_PREF attribute at the neighbour can of course overrule this rule. Rule 2 defines that the path with the least number of ASes is preferred, similar to the traditional shortest path routing [1].

Once the BGP decision process ends, the best path is installed in the forwarding table and exported to other routers. Besides the manipulation of some of the attributes by an announcing AS, routes can be suppressed (not announced at all) or aggregated.

BGP is a vectoring protocol that only sends incremental updates, i.e. only when a currently used path or policy has changed. Therefore, BGP achieves a greater degree of scalability than link state protocols making it better suited for the inter-domain scenario.

Although BGP is a very simple protocol, running it at such large scale raises scalability and stability problems. BGPs stability is affected by path exploration during BGPs convergence and also by some anomalies (e.g. routing loops that appear due to misconfigurations or bugs). It has been shown in [22] that during BGP convergence, triggered by a withdrawal or link failure, BGP faces temporary packet loss, even though a policy compliant path to the destination might still exist. The main reason for such problems

is the extreme flexibility in setting policies to define the best paths. This combined with the independent nature of the ASes can lead to conflicting routing decisions. Another reason is the fact that BGP has a single path nature, meaning that all the traffic control objectives of the ASes have to be obtained by manipulating the routing state by using BGPs flexibility. This produces a complex system with behaviour sometimes difficult to predict.

2.1.3 Multipath Routing

With the objective of exploiting the resources of the network, multipath routing techniques have been proposed to provide multiple alternative paths between source-destination pairs. These techniques have a potential to aggregate bandwidth on various paths, as well as produce a variety of benefits, such as fault tolerance, higher bandwidth and better network utilization [23].

Multipath can also enhance the network security providing privacy features: take the examples of the man-in-the-middle and sniffers attacks. They both need a single path for a particular interception point and with multipath these attacks are harder since they need to be located along the multiple paths. Denial-of-service attacks can also be avoided, given that attacking any link would imply that the traffic will move to alternative physical paths that compose the logical path, making logical paths more robust [30][37].

It can also improve the protocols convergence problem after a link failure or route withdrawal

In this section we discuss several proposal for multipath routing we start by intra-domain multipath.

2.1.3.1 Intra-domain Multipath Routing

Although extensive research on multipath routing has been done, in practice it is still not widely deployed. Research has focused mostly on extending intra-domain routing proto-

cols for multipath support [4] [28] [38].

Protocols like OSPF, RIP and several other protocols can learn several short paths to a given destination based on the protocol metrics. Therefore the most well known solution for Intra-domain multipath is Equal-Cost Multi-Path routing (ECMP) routing.

ECMP is a routing technique for forwarding packets over multiple paths of equal cost, i.e. tied in routing metric calculations, to a single destination. By load-balancing traffic over multiple paths typically using simple round-robin distribution it provides increase in the bandwidth. In general provides a better use of network resources and a better resilience than normal routing solution, however sometimes it may not provide enough path diversity [19].

In order to overcome the path diversity problem another technique was proposed, the Unequal Cost Multi-Path, which implies using some other routes in addition to the shortest ones. However, these new routes do not guarantee *loop free* networks unless protocols use *loop free* conditions [30].

Multipath Routing in MPLS Networks

There have been several approaches on applying multipath routing to MPLS networks. These works have the common goal of solving the traffic engineering problems using multipath Label-switched paths (LSPs). LSPs are pipes where the traffic is conveyed across the MPLS domain. In MPLS networks multiple LSPs can be established between MPLS nodes to enhance the network performance and Quality of Service (QoS). These approaches are tightly related to load balancing, traffic splitting and fault tolerance [23].

In case of failure, it may be desirable to reroute LSPs regardless of their bandwidth and other constraints. Re-routing in MPLS networks is important because of the explicit paths and re-routing packets around failure can be done faster by pre-planning backup LSPs.

2.1.3.2 Inter-domain Multipath Routing

The research proposals for inter-domain multipath can be classified as proposals that are, extensions or slight modifications to the current BGP protocol (e.g. MIRO [39], STAMP [24], R-BGP [21]) and proposals for an entirely new routing protocol (e.g. NIRA [40], Pathlet routing [12], Path Splicing [26]). We will briefly describe two proposals that extend BGP, BGP multipath extensions, MIRO and one clean slate approach, NIRA.

BGP Multipath

Multipath routing might be the solution during BGP convergence and a series of other benefits aforementioned (in the beginning of section 2.1.3) can also be brought to BGP, although it has to advertise more routes apart from the best one.

The authors in [33] proposed an extension to BGP to overcome the problem of not being able to advertise multiple routes. However, the impact of these methods in BGP scalability is unclear, since advertising more routes means having more entities in the tables, as well as more exchanged messages, creating overhead. Also the definition of which routes to be announced is critical, since this can enhance the problem of inconsistent behaviour due to conflicting path choices in the several ASes [1].

Multi-path Inter-domain ROuting

In Multi-path Inter-domain ROuting (MIRO), the default routes are learned through the BGP protocol, and pairs of domains can negotiate additional paths. When an AS wants alternative paths to a given destination, it requests its direct neighbour ASes for the alternative paths they have (recall that ASes learn many paths to each destination, but choose to use and advertise only one based on policies). If the AS does not obtain the appropriate path, it can request its neighbours to request their neighbours and so on. When a suitable path is discovered, the ASes establish a tunnel between them identified by a tunnel ID [39].

Costumers that want to forward packets on the new path have to notify the AS, which then encapsulates these packets in an extra IP header with the destination and tunnel ID. When a neighbour AS receives these packets it might de-encapsulate them and then

forward them through a regular route to the destination if no alternative route from the neighbour to the destination exists. So on the alternate route some parts of the traffic may use destination-based forwarding while other parts use the tunnel [11].

The protocol provides flexibility for path selection, but with the cost of path negotiation (with a latency that can be excessive for reliability purposes) and forwarding overhead.

A New Internet Routing Architecture

A New Internet Routing Architecture (NIRA) is an architecture proposal for inter-domain routing where each host has as partial graph knowledge of the AS networks to which it is connected to. This partial graph contains all connections of the AS up to the core level (ASes with no providers). With this information the host chooses routes or possible multiple routes to the destination. The system uses a mapping service called Name-to-Route Lookup Service (NRLS) to retrieve destination's addresses, since hosts only know the map to the core.

NIRA also requires use of IPv6 and the addressing to be hierarchical, so packets are forwarded based on the source and destination addressees in a hierarchical sense: first upwards on the user's up-graph and then downwards on the destination's up-graph.

NIRA is heavily based on the provider-customer hierarchy, allowing an AS to choose its own set of hierarchies but does not deal well with peering links [1] [10].

2.1.3.3 Multipath Routing Discussion

The different protocols discussed above, give efficient solutions for multipath and ways to minimize delay and increase throughput.

In the intra-domain routing environment there are different solutions that can be applied, such as ECMP for metric-based routing or LSP for MPLS networks. The way to affect the overall traffic flow of traditional metric-based protocols is to adjust the link metrics over a network. However, sometimes this adjustment may be impossible.

In the inter-domain routing environment, the conditions are more complex because most of the different existing proposals imply changes in the inter-domain technology based on BGP (NIRA case), or are not so good in path diversity (MIRO case).

Based on the Internet routing and the latest trends of the studied protocols, it is interesting to think of multipath policy routing protocol with a simple destination-based hop-by-hop forwarding, because it would fit both inter and intra-domain scenarios and should improve some of the problems already mentioned.

2.2 Routing Models

Section 2.1 gave a brief overview on today's routing protocols. This section gives a sense on how algebras appear in the routing context, as well as their importance in the study and development of new routing protocols. Some linear algebra concepts are introduced to familiarize the reader with the terms used in the algebraic routing models used in this dissertation.

For many years now, linear algebras have been widely used to solve several scientific problems. The use of algebras brought very interesting insights, and ever since routing protocols were first being designed, there has been a sense that a more modular approach would serve better than a pure algorithmic approach.

2.2.1 Algebraic Basis

Traditional algebraic structures such as groups, fields, or rings do not provide the most suitable tools for problem solving and modelling in specific areas. More primitive structures can be of interest. A typical example is the arithmetic case, modelled by the algebraic structure,

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

composed by the set of the non-negative integers endowed with the ordinary multiplica-

tion and addition. It does not contain the properties of a field or a ring, and it still is a structure of interest for several problems. Such structures, or even some more primitive ones, can be of use for the modelling of routing. The reason is that the operations of composing and selecting paths in routing generally do not possess, all the properties imposed in groups, fields or rings [13].

One example of a routing problem is the well-known modelling of the *shortest path* problem in a graph, in which the usual algebraic solution is non-linear. Nevertheless, it can be written as linear equations in the algebraic structure,

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

i.e. the set of real numbers endowed with the operation Min (minimum of two numbers) and the addition. Through the study of the properties of such an algebraic structure, a more generic one may be considered,

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

formed by a set E endowed with two binary operations $\oplus$ and $\otimes$. This structure is referred to as a *bisemigroup*. In the particular case where the operation $\oplus$ induces an order in the set E, it is called *ordered semigroup* structure with the form,

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

where the order $\preceq$ is given by $\oplus$ as: $a \preceq b$ if $a \oplus b = a$. Some examples of problems that can be modelled by such structures are:

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

To describe traditional routing protocol behaviours in a very clean and precise way, algebraic structures like the previous may be used. Understanding under which conditions a

routing protocol can operate correctly is fundamental to designers of new protocols and routing solutions.

The *correctness* of a routing protocol is related to how a set of stable routes is calculated in finite time and whether there are forwarding loops. Therefore, whenever it is referred that a routing protocol is correct it means that it complies with the following definition in [1]:

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 using *destination-based hop-by-hop* forwarding).

The use of a theoretical model and associated verification techniques confirms if a given protocol is *correct*.

2.2.2 Theoretical Model

In order to model the routing protocol, it is necessary to model three fundamental aspects: the set of link attributes; the process of obtaining paths by appending links and the process of deciding/selecting the paths to be used.

The primitive algebraic structures called *bisemigroups* (or *ordered semigroups* if we use the order theory terminology) are used instead of *semirings* since they need less properties to be verified and are therefore applicable to a wider set of routing models. It is formed by a set S endowed with two associative binary operations $\oplus$ and $\otimes$:

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

The set S models link and path attributes, it contains the special elements, $\overline{0}$ that models a non-existent or invalid path, and $\overline{1}$ that models a trivial path (the path to the node

itself) containing no links. The $\oplus$ operation models the path selection and in order to do so it should be idempotent, that is $\forall a \in S\ a \oplus a = a$, it can be applied multiple times with the same elements without changing the result. The $\otimes$ operation models the calculation of paths (path composition operation), expressing how the attributes of a path change with the addition of links. It is applied between a path value $p \in S$(starting with the trivial path) and a link value $l \in S$ resulting in another path value also from the set S.
A *canonical order* $\preceq$ for an idempotent $\oplus$ operation, can be defined as:

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

This can model the path selection operation as a preference order of path values in S, where $b \preceq a$ means that b is preferred over or equal to a (in the same order, $b \prec a$ means that b is strictly preferred to a). The order is bounded between the trivial path (the most preferred) and the invalid path (the least preferred) meaning $a \in S\ \overline{1} \prec a \prec \overline{0}$ [1] [15],

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

Choosing between *ordered semigroups* or *bisemigroups* for the theoretical model is simply a matter of considering the path selection as the result of the relative position of paths in an order, or as the result of a binary operation over two paths weights.

2.2.3 Correctness Verification

The first step to verify the *correctness* of a routing protocol is to check if it will converge to a stable set of best paths for each destination in finite time.

The network is modelled by a directed graph $G(V, E)$, where V is a set of vertices and E is a set of edges. Considering an attribute/weight function $w(i, j)$ that maps $E \to S$, thus providing values to edges. A path is a sequence of edges, $P = v_1, v_2, \cdots, v_k$ and its weight is given by $w(P) = w(v_1, v_2) \otimes w(v_2, v_3) \otimes \cdots \otimes w(v_{k-1}, v_k)$. The routing problem is to calculate for all pairs $(i, j) \in V$ the weight of $P(i, j)$ which defines all paths between i and j, and by using $\oplus$, the set $O(i, j)$ of the smallest weight paths is obtained (best paths):

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

The solution can be obtained by taking the matrix version of operations $\otimes$ and $\oplus$ and solving one of two matrix equations, the left (2.1) equation, or the right (2.2) equation [13] [3]:

$$L = (A \otimes L) \oplus I \tag{2.1}$$

$$R = (R \otimes A) \oplus I \tag{2.2}$$

where A is the adjacency matrix of the network graph:

$$A(i,j) = \begin{cases} w(i,j) & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

I is the identity matrix, and R,L matrices have the smallest weight of paths values $R(i,j)L(i,j)$ between nodes i and j, given the set of the smallest weight paths of the neighbours q of source i:

$$L(i,j) = \bigoplus_{q \in V} A(i,q) \otimes L(q,j) \tag{2.4}$$

$$R(i,j) = \bigoplus_{q \in V} R(i,q) \otimes A(q,j) \tag{2.5}$$

The difference between the left and right equations is in the order of the path weight calculation where for the left case (2.4) starts from the destination to source and for the right case (2.5) from the source to destination.

In both cases the solution can be found by calculating the closure matrix of the adjacency matrix A denoted by A^* via matrix iteration by successively applying $\otimes$ to A and minimizing the result for each element of the matrix according to the preference order $\preceq$ [13] [3]. If A^* exists then, it is assured that a routing protocol will converge to a stable routing solution. The existence of A^* depends on the properties of the *ordered semigroup*.

2.2.4 Condition of Convergence

One condition of convergence to a stable routing solution is monotonicity for *ordered semigroups*. *Monotonicity* is defined as [13]:

Definition 2.1. An *ordered semigroup* $(S, \preceq, \otimes)$ is monotone if

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

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

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

The best path is the lowest path in the $\preceq$ order. Naturally 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 [3]. In this case the solution is a global optimum solution and it can be obtained through matrix iteration by consecutively adding links and choosing the more $\preceq$ preferred paths between each (i, j) pair of vertices of G. The *Bellman-Ford* and *Dijkstra* algorithms [7] are other known methods to compute the solution for respectively the right routing problem equation (equation 2.2) and the left routing problem equation (equation 2.1) that can be related with matrix iteration [13] [36].

An important issue is the *strictness* of monotony. Definition 2.1 is the *non-strict* case which assures that A^* exists and that a solution will converge to it, however it does not assure convergence in finite time. The reason is that in a *non-strict* monotone algebra the $\otimes$ addition of a link can result in equally preferred ($\simeq$) weights when their sub-paths had different weights:

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

This also means that preference can be maintained with the addition of links to a path. In

a *non-strictly* monotone algebra there may occur cycles in the network where an infinite number of paths have the minimal weight (by consecutively passing around the cycle). This means that an infinite number of paths have the same preference. The consequence is that, the convergence to the routing solution is not obtained in finite time, especially if we consider multipath (meaning that we need to maintain all equally preferred paths). This poses a practical problem when implementing a protocol modelled by such algebra. So, to assure convergence in finite time, monotony should be strict:

$$\forall a, b, c \in S \quad a \prec b \Rightarrow (c \otimes a) \prec (c \otimes b)$$

Monotonicity is very hard to maintain in policy based protocols. One of the reasons for this is that the *forbiddingness* of paths due to policy easily breaks monotony [15] [34]. This means that policy based protocols need to be modelled differently. The first approaches to model policy based protocols were proposals to model the inter-domain policy based protocol, BGP. One of the first models was a graphical theoretical model called Stable Paths Problem (SPP).

2.2.4.1 Stable Paths Problem

The SPP framework [16] was the first approach for modelling policy-rich routing protocols. It considered a formalization of BGP based on a simple graph-theoretical approach and represents a specific set of policies. 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 to a given origin and a rank assigned on its permitted paths that corresponds to its level of preference. A solution to the Stable Paths Problem does not represent a global maximum, but rather an equilibrium point in which each node is assigned its local maximum.

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

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*.

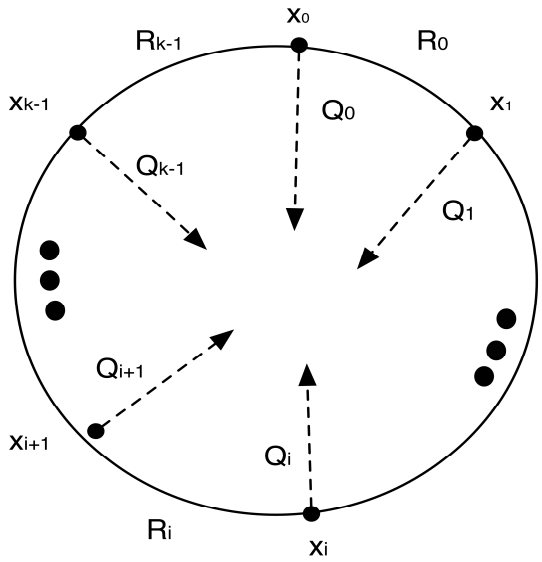


Figure 2.1: Generic dispute wheel [1]

A *dispute wheel* occurs when nodes prefer the path using the next node in a cycle instead of a direct path to the destination that does not travel around the cycle. Figure 2.1 shows the nodes $x_0, ..., x_{k-1}, x_k{=}x_0$, the paths Q_i the destination and the paths between nodes where R_i is a path from x_i to x_{i+1}. We have a disputed wheel if the path composed $R_i + Q_{i+1}$ is more preferred than the path Q_i for node x_i. 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.

2.2.4.2 Path Vector Algebras

In [34] the author proposes another theoretical model for inter-domain routing. It follows an algebraic perspective and introduces what was called path vector algebras, providing the needed properties for the correct operation of policy based protocols. The algebraic theory can be considered a generalization of the shortest path routing performed by the same author [35].

Each link has a label applied to it and the total value of the path is obtained by applying the binary operation $\otimes$ to the consecutive links that form the path. The $\preceq$ order models the choice of the best path completing the model of the protocol. The preference order places preferred paths in the lowest order. This is opposite to the SPP framework where the higher paths in the order were the most preferred ones.

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 [20] a translation between path algebras and SPP is provided. The absence of *free* cycles in the network is a sufficient condition that assures that the algorithm terminates finding a locally optimal solution. This brings us to more recent work that casted this work in to a pure algebraic setting.

2.2.4.3 Pure Algebraic setting

Recent work shows that a routing solution can still be obtained in finite time if the $\otimes$ operation is *increasing*. The *increasing* property is defined as:

Definition 2.2. An *ordered semigroup* $(S, \preceq, \otimes)$ is *increasing* if [1]

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

This means that the addition of a link always decreases path preference, increasing its

position in the order $\preceq$. This does not assure anymore that every subpath of a best path is also a best path. However, a solution for the left and for the right equations (2.1) (2.2) 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 kinds of proof can be found in [13] [3] [34] [14] and the author in [20] showed that if the algebra is *strictly increasing* then *non free* cycles never occur in the network. Since multipath is an interesting feature to deal with many of the issues identified in the first section of this chapter, we need to define a model for multipath routing.

2.2.4.4 Modelling Multipath

Multipath routing can be modelled by defining minimal set(minset[2]) of S [1]:

Definition 2.3. A minimal set is a subset $C \subset S$ with cardinality $|C|=n$ such that

$$C \;=\; min_{\preceq}(C) \;=\; \{x \in C \;|\, \forall y \in C\colon\; x \preceq y\}$$

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 $(\oplus', \otimes')$ to work with the minimal sets instead of the individual elements of S. The operator $\oplus'$ returns the most preferred paths out of the union of A and B. Likewise, $\otimes'$ 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 $M(S)$.

If an algebra is monotone the minimal set routing solution can be found and it is a global optimum [1], but as pointed out above, monotony is easily lost in policy routing. In the absence of monotony, it is sufficient to have a *strictly increasing* $\otimes$ operation to assure

[2] In this Dissertation we will use the terms "minimal set" and "minset" with the same meaning

convergence to a local optimal routing solution using multiple paths of equal or equivalent preference. Proof can be found in [17] as well as in [5] where the author finds a similar conclusion using a notion that relates the SPP framework with algebraic routing called policy relation.

In summary, the convergence conditions are the same for the single path case, and given that monotony is too restrictive for policy-based protocols the focus is on models that merely have the *strictly increasing* $\otimes$ operation property. A protocol with such a model has finite time convergence because adding a link always increases the path weight and therefore decreases the preference. So, preference around cycles always decreases and the protocol converges in finite time regardless of what cycles occur in the network.

The *strictly increasing* property is very restrictive for multipath. The reason is simple to understand: in a *strictly increasing* model only paths with an equal amount of links can be considered of equal weight and this reduces the amount of paths that have the same weight and can therefore be used simultaneously, thus defeating the purpose of a multipath protocol.

Considering Definition 4.1. the last step is checking if forwarding loops occur assuming destination-based hop-by-hop forwarding. Forwarding loops are caused by (independent) forwarding decisions that are not consistent. In a *non-monotone* $\otimes$ *non decreasing* model we have to distinguish equation 2.1, in which $\otimes$ is applied to the left, from equation 2.2 where $\otimes$ is applied to the right since in the absence of monotony their solutions can be different. It is well known that right local solutions can cause forwarding loops in destination-based hop-by-hop forwarding [36] because 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. When the calculation is done from destination to source (equation 2.1), forwarding loops may not occur since path weights are always dependent on the weights of the sub-paths from the next hops to the destination.

2.2.4.5 Multipath Forwarding in Non Decreasing Models

We have established that a routing solution will be calculated in finite time provided that *non free* cycles are not present. For single path routing this also means that forwarding loops do not occur in destination-based hop-by-hop forwarding if the solution is calculated by applying $\otimes$ to the left. Multipath routing introduces a new situation in forwarding that needs to be taken in consideration when evaluating if a cycle is *free*.

We derived the following method of checking for *non free* cycles from a method presented in [36], that was recast for the pure algebraic *ordered semigroup* model:

Definition 2.4. For each value $s \in S \setminus \overline{0}$ consider a set $L_s \subset S$, $L_s = \{l \in S |$ 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 [1].

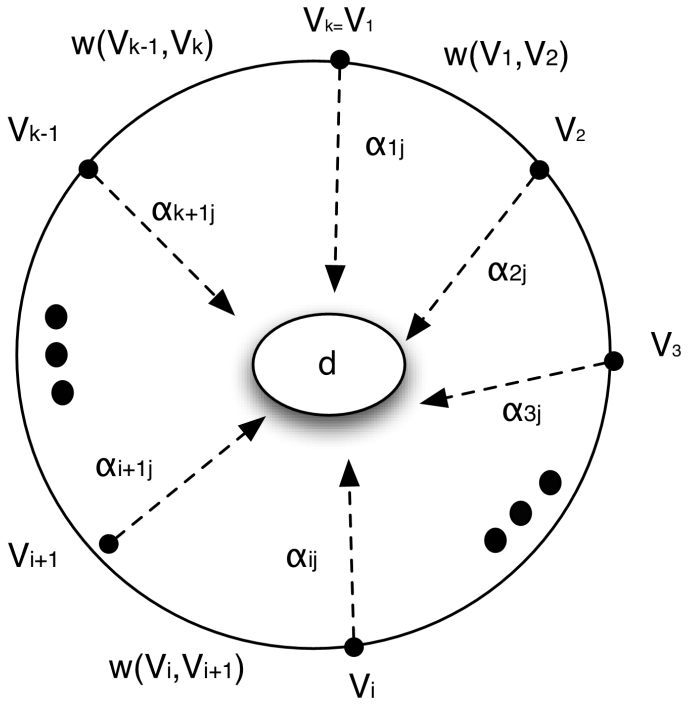


Figure 2.2: Free cycle illustration [1]

Consider an *ordered semigroup* with $S = \overline{1}, a, b, c, \cdots, \overline{0}$ and $\overline{1} \prec a \prec b \prec c \prec \cdots \prec \overline{0}$. An *ordered*

semigroup that is merely *non decreasing* has the $\otimes$ operation defined as $\forall x, y \in S$ if $x \prec y$ then $x \otimes y = y$ and $\forall z \in S\ z \otimes z = z$.

Now consider that in the cycle of figure 2.2 all links between V_{k-1} and V_3 are labelled with b in both directions forming a sub-path meaning $w(V_{k-1}, V_k) = w(V_k, V_{k-1}) = w(V_1, V_2) = w(V_2, V_1) = w(V_2, V_3) = w(V_3, V_2) = b$. Consider V_i (that does not belong to the sub-path) and the links from V_{k-1} to V_i and from V_3 to V_i have weight $w = a$ in the forward direction and $w = c$ in the reverse direction. If the method of definition 2.4 is applied to check if the cycle is *free*, the obtained result is three problematic sets $L_a = a$; $L_b = b$ and $L_c = c$. This means that with the above assignment of labels to the links the cycle is *free* since it has links with labels a, b and c. So, the cycle does not bring problems in terms of finite time convergence to a routing solution since paths around the cycle do not maintain preference.

However, in multipath with the *non decreasing* operation, forwarding loops can occur inside a sub-path that maintains preference. To see this consider node $V_k = V_1$; it has two equally preferred paths to V_i (clockwise and counter-clockwise) with the weight b since $b \otimes a = b$. The same happens with V_2. A forwarding loop can then occur between $V_k = V_1$ and V_2 with packets bouncing between the two nodes. Note that for this cycle this only happens for V_i. Another node outside of the subpath (different from V_i, and keeping $V_k = V_1$ or V_2 as the source) has one path with weight equal to b in one direction and another path equal to c in the other direction.

For a *non decreasing* $\otimes$ operation if more than three nodes are connected through links with the same label in both directions then two of the nodes can have equal weight paths in both directions of the cycle to some destination causing a forwarding loop. In order to assure multipath *loop free* forwarding the following statement was added to definition 2.4 [1].

A cycle is non free if three or more of its links are labelled with the same element l (belonging to one of the L_s sets) in both directions.

2.3 Summary

In summary we have seen that known algorithms can convergence to a multipath routing solution by variants of classical methods of linear algebra.

We started by seeing that a routing protocol modelled by a strictly monotonic *ordered semigroup* is ideal to assure convergence to a global optimum in finite time convergence but has problems for policy routing and multipath. The reason is that there are many constraints to consider two paths as equally preferred.

A second option was a simple monotonic *ordered semigroup* which also assures global optimum solution, but in this case convergence in finite time is not assured and forwarding loops are also possible. However, if we drop monotonicity and accept local optimum solutions, we get more algebraic models to solve routing problems.

Another option is an *ordered semigroup* not monotone with $\otimes$ *increasing* operation to provide a local optimum solution in finite time using the left local approach (applying from destination to source). This convergence using the $\otimes$ *increasing* operations is still not the best choice because hop count has too much influence in the paths weights, which is not a good for policy based routing.

The most interesting option is an *ordered semigroup* not monotone with the $\otimes$ *non decreasing* operation to provide a local optimum solution in finite time using the left local approach. The network has to be constrained in having all cycles *free* and cannot have more than three consecutive links label with symmetric policies connecting them (the same policy in both directions of the link). This solution for multipath routing is of interest since it allows a better approach to many of the issues discussed and can be used for intra and inter-domain routing. In that context achieving equal preference for the maximum possible number of paths is the final goal.

Chapter 3

Multipath Policy routing Protocol

3.1 Introduction

In the previous chapter (in section 2.2), algebraic models were reviewed covering the recent work on modelling routing protocols, and the necessary convergence conditions. So far we have seen that shortest path protocols can be modelled by *semirings* which have well-known properties that assure convergence in distributed routing with destination based hop-by-hop forwarding. However, as showed in section 2.2.4, policy routing models are different algebraic structures where some of the *semiring* properties are lost. The reason is that a policy is a richer concept than a simple metric defining a generic quality of a path/link, and the path preference depends on it.

In this chapter we present a model for a generic multipath policy routing protocol and provide its *correctness*. The model is built on three features that can be of interest for several scenarios, but that are more difficult to model than traditional shortest path protocols, they 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 this dissertation we implement a generic policy routing protocol featuring multipath proposed in [1], which is described in this chapter. The trade-offs involved in the construction of such protocols are evaluated through algebraic analysis. We start by having no restrictions other than the ones that the resulting algebraic model must fulfil, i.e.,complying with at least one of the convergence conditions stated in chapter 2, section 2.3.

3.2 Protocol Algebraic Model

The concept of hierarchy fits reasonably well in the way networks are organized. 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. Within the transport/backbone area some hierarchical levels can also be present. This 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, given the concept of hierarchy it is natural that we define a set of policies based on the relative position between two nodes in the network hierarchy. Paths are qualified with a weight depending on the direction of the hierarchy they follow. We can identify two main purposes to use policies:

- To set certain routing objectives, like allowing paths to be chosen by a certain attribute that may be unrelated to any measurable metric, regarded as a qualitative identification;
- To control which paths are known and can be used.

By using policies, several paths can end up having equivalent weights and behave similarly in the hierarchy, but can be different in all other aspects (e.g number of hops, links,

capacity, or bandwidth)

Since the protocol features multipath it should be modelled using a *minset* algebra defined over an ordered *semigroup* (Definition 2.3). It has been seen in previous chapter (section 2.2.4.4) that extending the ordered *semigroup* model using the *minset* operator to express the use of multiple paths has no impact in the required algebraic properties for the protocol to converge. Therefore, we model our protocol using an ordered *semigroup* and analyse its *correctness*. The *minset* operator can then be applied 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 $\preceq$. The set 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 [1].

Note that the way values are assigned to links creates a specific path distribution for each network graph with the consequences of having more or less path diversity. Also, the definition of the set of values in S and the way they are composed with $\otimes$ defines 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:

1) D_W - links in the downwards direction of the hierarchy.

2) U_W - links in the upwards direction.

3) S_W - links between nodes at the same level of the hierarchy.

A link between two nodes at different levels of the hierarchy is modelled by two directed edges in the graph: one in the upward direction (U_W) and another in the downward di-

rection (D_W). If the link connects nodes at the same level of the hierarchy (peer node) both directed edges are labelled with the same value (S_L).

Trivial paths are modelled by $\overline{1}$ corresponding to the highest preference possible and invalid paths which have the lowest preference are modelled by $\overline{0}$. So we have S= $\{\overline{1},\ D_W,\ S_L,\ U_W,\ \overline{0}\}$. and the path preference is modelled by the preference order $\prec$,

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

meaning that D_W paths have the highest preference followed by peer S_L paths and finally U_W paths have the lowest preference. So, going downwards in the hierarchy is more preferred than going upwards.

The elements of S and the order relation 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.

In order for the $\otimes$ operation to be *non decreasing* the link with the lowest preference in a path defines the total path weight. This allows for a great number of paths to have the same weight and preference thus enabling their simultaneous use. 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 downwards in the hierarchy (D_W) are more preferred than the ones that contain at least one link in the same level (S_L). The least preferred paths are the ones that contain at least one link going upwards in the hierarchy (U_W).

The $\otimes$ operation is defined in table 3.1:
Looking at table 3.1 we see that there are three cases with an invalid entry. The reason is related to the algebra being *non decreasing* and to the number of possible *non free* cycles

$\otimes$	$\overline{1}$	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

Table 3.1: $\otimes$ Operation. [1]

that cannot be present in the network.

Using definition 2.4 in chapter 2 to analyse the possible *non free* cycles, we start by defining the sets L_s for each $s \in S \backslash \overline{0}$.

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). In this case L_{U_W} contains all possible link values, which causes all cycles to be *non free*, given that the $\otimes$ operation is *non decreasing*. This means that the network would have to be acyclic and therefore without multipath.

Take the example of disjoint paths leaving from a source at a high level in the hierarchy towards a node at a lower level, and from there to the destination. All of the disjoint paths could not exist simultaneously because they imply a cycle in the network in which all links belong to L_{U_W}.

A way to solve it is to change the algebra such that some of the L_S sets have fewer

elements, as showed in table 3.1. By invalidating paths the defined L_S sets are now the following [1]:

- $L_{\overline{1}}$ and L_{D_W} remain equal
- L_{S_L}=$\{S_L\}$ given, that now $\overline{0}$=$D_W \otimes S_L$
- L_{U_W}=$\{U_W\}$ given, that now $\overline{0}$=$D_W \otimes U_W$ and $\overline{0}$=$S_L \otimes U_W$

The invalid combinations of links resulting in weight $\overline{0}$ exist to reduce the number of possible cycles. For instance, the L_{S_L} set, by invalidating the result $\overline{0}$=$D_W \otimes S_L$ only cycles where all links have the same label are *non free*.

Let us examine briefly each of the possible *non free* cycles. The L_{D_W} and L_{U_W} cases actually reduce to one since labels D_W and U_W are only set in one direction of the link with the complimentary label being applied in the reverse direction. It means that a cycle has all links with label D_W in one direction and all links with label U_W in the other. Such a cycle cannot exist in a well-formed hierarchy, or at least it is of no interest. Concerning the set L_{S_L} there are two possible problems,
The first problem concerns the inability to achieve the multipath routing solution in finite time. Consider the figure 3.1, with links between $a_1, a_2, a_3, \ldots, a_{n-1}, a_n$=$a_1$ all of type S_L. Consider that the destination is node d and link L_{a_2-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 not to converge in limited time because a valid path with minimal weight can be constructed by endlessly circulating around the cycle. The second problem concerns forwarding loops that can also occur in such cycles. This happens because all a_i nodes in the cycle (except a_2) have paths to d with S_L weight by following any of the directions of the cycle. 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 a 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. Even a simple link between two of these nodes can form a cycle due to path being extended back and forth between the nodes without the preference being decreased.

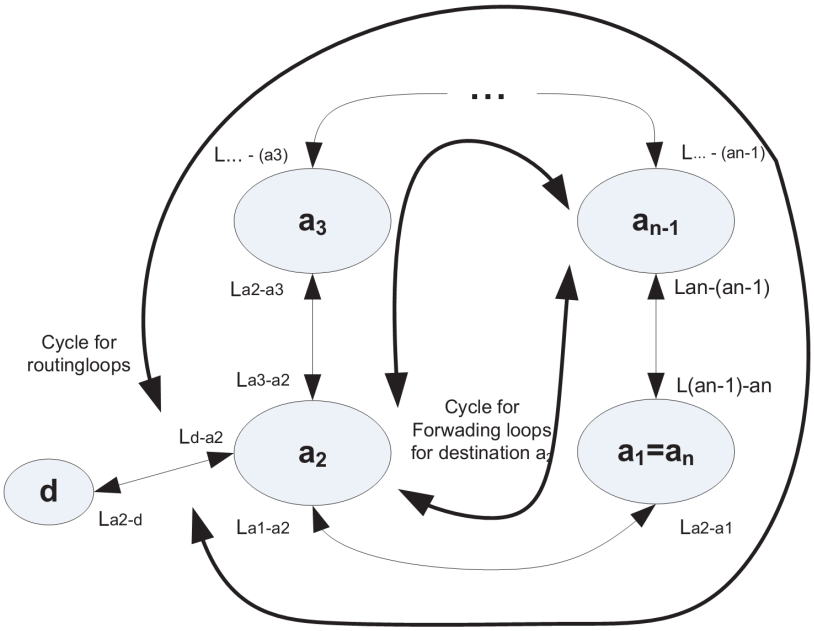


Figure 3.1: Free cycle illustration. [1]

The forwarding loop problem in these cycles is related with the label S_L being applied in both directions of the links. As showed forwarding loops can occur if three or more links in a cycle have links with L_{S_L} and only with L_{S_L}, since that is the only set whose elements $\{S_L\}$ are applied symmetrically to links. As an example 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. We have the situation described for V_i in figure 2.2. The path calculation stage is now bounded since the cycle is no longer made of only S_L links, e.g. when a path is 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 $\overline{0}{=}D_W \otimes S_L$, the path stops in a_1. So, the operation of path calculation stage is no longer infinite. However forwarding loops still occur and therefore the cycle is still *non free* according to our extension to definition 2.4.

Non free cycles like this are quite restrictive since they imply that no consecutive S_L links occur in the network (as stated in the end of chapter 2: two or more consecutive S_L). A possible solution would be to invalidate paths with more than one consecutive S_L link

($\overline{0}$=$S_{L_W} \otimes S_L$). In fact, protocols with a *non decreasing* ordered *semigroup* model cannot allow consecutive links of equal value of policy in both directions of a link to be formed. This is very restrictive and prevents different paths at the same level of a hierarchy to be fully used, thus defeating the multipath characteristic.

The next subsection describes a solution for the problem that allows more flexibility.

3.3 Forwarding Loop *free* Paths with Multiple S_L Links

The solution we adopted was to change the model so that adding a S_L link has consequences in terms of preference depending on the direction of the link. For this, a secondary label was added to S_L links creating a sense of direction as well as a secondary preference. This way the same level links are no longer labelled equally in both directions and have different preferences according to the different directions. This solution solves both the path calculation problem and the forwarding loops since paths do not have the same preference in both directions of the cycle any more.

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) [13][3][18].

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

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

The secondary *ordered semigroup* is $(T, \preceq_T, \otimes_T)$ with the set

$$T=\{\overline{0},\ \ L,\ \ R,\ \ \emptyset\} \tag{3.1}$$

The $\otimes_T$ operation is shown in table 3.2.

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

Table 3.2: $\otimes_T$ Operation. [1]

The preference relation on the secondary label is given by,

$$\emptyset \prec_T R \prec_T L \prec_T \overline{0} \tag{3.2}$$

The idea is that amongst paths with consecutive S_L links, the preference depends on the secondary label, L or R, 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, or not used at all due to the preference order. It might be interesting to have a policy where a path in the same level can only reach the direct neighbour or nodes below it. The motivation is to drive "local" traffic through local links instead of going up in the hierarchy (something hard to accomplish with shortest path routing protocols) but only for nearby nodes so that the higher diversity of paths through the higher hierarchies are still usable for more distant nodes. In order to allow this flexibility S_L links were subdivided into two groups in the primary algebra: one that limits the number of S_L links on a path to one, labelled S_{L1}, and another that allows paths to have multiple S_L links labelled S_{L2} (which have then a secondary label, L and R). The set S is now

$$S=\{\overline{0},\ U_W, S_{L1},\ S_{L2},\ D_W,\ \overline{1}\} \tag{3.3}$$

and the preference order $\preceq_S$ is given by,

$$\overline{1} \prec_S D_W \prec_S S_{L1} \prec_S S_{L2} \prec_S U_W \prec_S \overline{0} \tag{3.4}$$

The $\otimes_S$ operation is showed in table 3.3.

$\otimes_S$	$\overline{1}$	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

Table 3.3: $\otimes_S$ Operation [1]

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

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

where [1]

$$(s_1, t_1) \preceq (s_2, t_2) \iff s_1 \prec_S s_2 \vee (s_1 =_S s_2 =_S S_{L2} \wedge (t_1 \preceq_T t_2))$$

and

$$(s_1, t_1) \otimes (s_2, t_2) \;=\; (s_1 \otimes_S s_2,\;\; t_1 \otimes_T t_2) \tag{3.5}$$

This means that paths have the value $\emptyset$ in T until they pass through one S_{L2} link. Then they might have 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.

This solves the problems of equal preferences being maintained over consecutive S_{L2} links. If all links in the cycle belong to sets L_1=$\{(S_{L2}, R)\}$ or L_2=$\{(S_{L2}, L)\}$ with $L_1, L_2 \subset (S \overrightarrow{\times} T)$, *non free* cycle may still occur. In order to avoid such cycles, left-right orientations must be defined in a plane, leaving at least one node on the opposite side of the other.

Take for example the cycles in figure 3.2 where all links are labelled S_{L2}. The labelling of the links can be done in two possible ways.

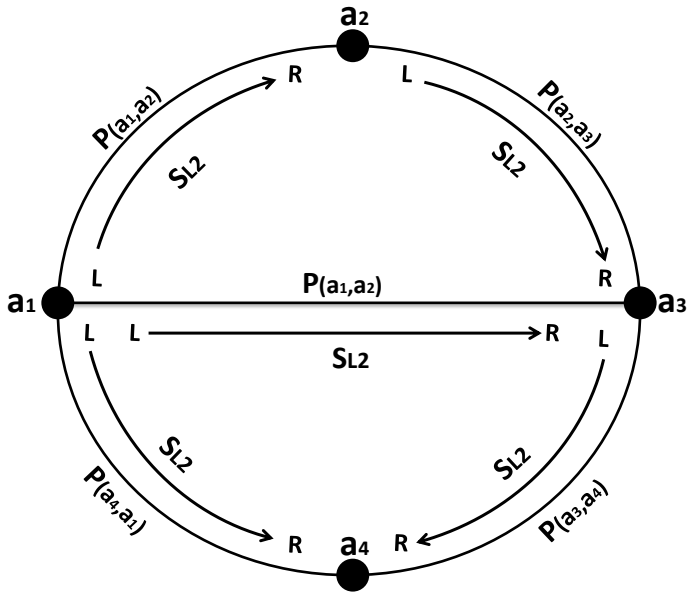


Figure 3.2: S_{L2} cycle free labelling

The first is following the clockwise direction of a cycle, consider the outside cycle: $P(a_1, a_2)$= $P(a_2, a_3)$=$P(a_3, a_4)$=(S_{L2}, R) all these links connect nodes to the right, the opposite link is $P(a_4, a_1)$=$(S_L 2, L)$. Note that all links having label R in one direction have label L in the other. The cycle is free since going clockwise we have $P(a_4, a_1) \notin L_1$. The same reasoning applies to the cycle in the lower half of the circle, as for the cycle in the top half because $P(a_3, a_1) \notin L_1$.

The second way is considering the anticlockwise direction. In this case we have: $P(a_4, a_3)$= $P(a_3, a_2)$=$P(a_2, a_1)$=(S_{L2}, L) and $P(a_1, a_4)$=(S_{L2}, R). Once again, with such labelling the cycle is free since $P(a_1, a_4) \notin L_2$.

3.4 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 $(S \overrightarrow{\times} T)$ [1].

The lexicographic product is studied in [18], and theorems are derived that relate the properties of the individual algebras to the ones of their lexicographic product. In our case both S and T are *not strictly* monotone and *non decreasing*. The first step to verify is if their product is also *non decreasing*. According to [18] the lexicographic product is *non decreasing*(ND) if

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

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

The next step is to check which are *non free* cycles. Analysing the cycles according to definition 2.4 the problematic case is the set originally defined as L_{S_L} since the introduction of the secondary algebra only affects S_L links. 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 $S_{L1} = l \otimes_S S_{L1}$;
- $L_{S_{L2}}$=$\{S_{L2}\}$ since $S_{L2} = S_{L2} \otimes_S S_{L2}$

For the total $(S \overrightarrow{\times} T)$ algebra the later set is divided in two:

- $L_{S_{L2} \overrightarrow{\times} L}$=$\{(S_{L2}, L)\}$ 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. So, the protocol allows cycles made of S_{L2} links to exist provided that at least one link is labelled in T in the opposite direction of the others. In conclusion the protocol is *correct* if the network fulfils the following two conditions:

1) No cycles exist that do not respect the up/down hierarchy.

2) S_{L2} edges are labelled in T such that it is impossible to travel around a cycle following only (S_{L2}, L) or only (S_{L2}, R) links.

This concludes our proof.

3.5 Protocol Implementation Model

The contribution of this dissertation is the implementation of the protocol model presented in the previous sections. This subsection presents the algebraic model of the algorithm used to calculate the routing solution. The first step in the algorithm is to obtain a representation of the network topology.

The adjacency matrix A is the algebraic representation of a network graph/topology and is always a square matrix. Consider $P(i,j)$ the set of all paths between a source i and destination j. Their weights are given by $w(P)$ and belong to the set S as shown in equation 3.3. This representation of matrix A is shown in equation 2.3.

The identity matrix I has the same size of matrix A, and the main diagonal only contains the trivial path with value $(\overline{1}, \emptyset)$ (representing the most preferred) and the remaining signatures are labelled with unreachable value $(\overline{0}, \emptyset)$ (representing the less preferred),

$$I = \begin{bmatrix} (\overline{1},\emptyset) & (\overline{0},\emptyset) & \cdots & (\overline{0},\emptyset) \\ (\overline{0},\emptyset) & (\overline{1},\emptyset) & \cdots & (\overline{0},\emptyset) \\ \vdots & \vdots & \ddots & \vdots \\ (\overline{0},\emptyset) & (\overline{0},\emptyset) & \cdots & (\overline{1},\emptyset) \end{bmatrix}$$

The routing solution can be found by solving the matrix equation 2.1, also called left routing problem equation, obtained through the closure matrix of adjacency matrix A denoted by A^*. Matrix A^* may be obtained via matrix iteration [13], corresponding the following matrix equation,

$$A^{k+1} \;=\; A \otimes A^k \;\text{ for }\; \mathrm{k}\in\mathbb{N} \tag{3.6}$$

Identity matrix I is not present, it is only used in the first iteration, meaning we start

with the identify matrix I and left $\otimes$ multiplying A recursively. A^* is obtained if equation 3.6, reaches a convergence solution in a finite number of iterations, i.e. when it verifies the condition,

$$A^k = A^{k+1} \text{ for } k \in \mathbb{N} \tag{3.7}$$

The following examples will help to understand how the matrix multiplication works. Consider now the graph in figure 3.3 with two levels of hierarchy.

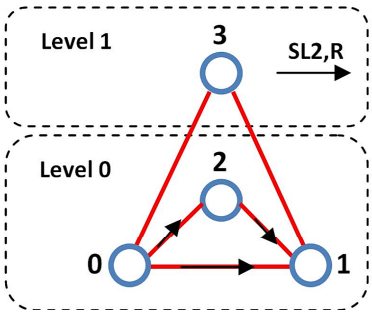


Figure 3.3: simple hierarchical graph

In the example of figure 3.3, nodes 0 and 1 rise in hierarchy (U_W) to reach node 3, and on the other way node 3 descend in hierarchy (D_W) to reach both nodes. Nodes 0, 1 and 2 are at same level (S_{L2}), this would cause a cycle as mention in definition 2.4 and to make cycles free a second algebra was inserted. Let us assume the direction left to right as shown in the figure (node 1 is to the right of node 0). The adjacency matrix is,

$$A = \begin{bmatrix} (\overline{1},\emptyset) & (S_{L2},R) & (S_{L2},R) & (U_W,\emptyset) \\ (S_{L2},L) & (\overline{1},\emptyset) & (S_{L2},R) & (U_W,\emptyset) \\ (S_{L2},L) & (S_{L2},R) & (\overline{1},\emptyset) & (\overline{0},\emptyset) \\ (D_W,\emptyset) & (D_W,\emptyset) & (\overline{0},\emptyset) & (\overline{1},\emptyset) \end{bmatrix}$$

The decision of S_{L2} policy on links was made in order to show the multipath feature later on.

Now, to find the routing solution, equation 3.6 is used. The most perceptible form to calculate the solution is given by equation 2.4. Take for instance the calculation for the position (1,2) of a generic matrix A with n elements.

$$a_{12}^{k+1} = (a_{11} \otimes a_{12}^{k}) \oplus (a_{12} \otimes a_{22}^{k}) \oplus \cdots \oplus (a_{1n} \otimes a_{n2}^{k})$$

This means that to obtain the most preferred weight for a path with k edges from origin 1 to destination 2 (a_{12}^{k+1}) it is necessary to compose all possible paths to the destination using the $\otimes$ operation Path Composition and then choosing the most preferred values using the $\oplus$ operation Path Selection.

The Path Composition operation composes various paths extending one path at a time. For the aforementioned example the first path is composed by the origin (1) and a next hop (n); and the second is the path from next hop (n) has to the destination (2). Since the equation 3.6 solves the multipath problem, in the Path Selection operation it might appear several equally preferred next hops.

Let's take the example from the topology of figure 3.3, using matrix A to calculate the value of the path from node 0 to node 1 ($A_{(0,1)}^{2}$), for k=1 (second iteration) where both nodes are on the same level,

$$\begin{aligned} A_{(0,1)}^{2} &= (A_{(0,0)} \otimes A_{(0,1)}) \oplus (A_{(0,1)}\ A_{(1,1)}) \oplus (A_{(0,2)} \otimes A_{(2,1)}) \oplus (A_{(0,3)} \otimes A_{(3,1)}) \\ &= ((\overline{1}, \emptyset) \otimes (S_{L2}, R)) \oplus ((S_{L2}, R) \otimes (\overline{1}, \emptyset)) \oplus ((S_{L2}, R) \otimes (S_{L2}, R)) \oplus ((U_W, \emptyset) \otimes (D_W, \emptyset)) \\ &= (S_{L2}, R) \oplus (S_{L2}, R) \oplus (S_{L2}, R) \oplus (\overline{0}, \emptyset) \end{aligned}$$

Sorting by preference using the Path Selection operation we are left with

$$A_{(0,1)}^{2} = (S_{L2}, R) \prec (\overline{1}, \emptyset)$$

The preferred paths result from the extensions ($\oplus$ addition) of the paths are from nodes (0,1,2), but since node 0 cannot be the next hop (to itself), so the next hops for destination 1 are in fact {1,2} (nodes 1 and 2). This shows that there are two paths able to reach the destination, allowing multipath in this case. Process 3.6 is repeated to solve the entire matrix until it reaches stop condition from equation 3.7. The solution for the example of

figure 3.3 is,

$$A^2 = \begin{bmatrix} (\overline{1},\emptyset) & (S_{L2},R) & (S_{L2},R) & (U_W,\emptyset) \\ (S_{L2},L) & (\overline{1},\emptyset) & (S_{L2},R) & (U_W,\emptyset) \\ (S_{L2},L) & (S_{L2},R) & (\overline{1},\emptyset) & (\overline{0},\emptyset) \\ (D_W,\emptyset) & (D_W,\emptyset) & (\overline{0},\emptyset) & (\overline{1},\emptyset) \end{bmatrix}$$

The first iteration results in $A = A^2$, therefore finding the topology solution. We can see that although the topology is small there is no valid path between nodes 2 and 3, which is correct if we analyse table 3.3. One reason for an invalid entry is related to possible *non free* cycles that cannot be present in the network.

The next hop table for the topology is,

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} - & \{1,2\} & 2 & 3 \\ \{0,2\} & - & \{0,2\} & 3 \\ 0 & 1 & - & \{\emptyset\} \\ 0 & 1 & \{\emptyset\} & - \end{bmatrix} \end{array}$$

This table is filled during the Path Selection operation. The value $\emptyset$ corresponds to an empty value, which means that there are no next hops to the destination.

This iteration matrix algorithm is related to an asynchronous distributed version of the *Bellman-Ford* algorithm that calculates the routing solutions by applying $\otimes$ and $\prec$. The algorithm itself forbids repetitions of nodes in one path [17]. So it is also possible to find a routing solution with a distributed algorithm that does not require a previous knowledge of the matrix A.

3.6 Protocol Characteristics

From the protocol model there are some conclusions that can be drawn concerning these types of protocols. A first aspect is that there is 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. To increase the possibility to have equally preferred paths the $\otimes$ operation as to be merely *non decreasing*, and this implies that the network has to be restricted to not having *non free* cycles. In order to assure *correctness* without posing too many restrictions on network topology, some paths had to be considered invalid or else too many cycles would be *non free*. Also, although there might be multiple paths they must have equivalent preferences in the 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 the algebraic model and the defined preference order. 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. 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, which is an advantage for multipath.

A particular impact on the amount of available paths for a given node is made by same level links. 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. This makes 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 even higher level

nodes). 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.

The next chapter describes the implementation of the algorithm presented above to calculate the routing solution for the protocol model given a network graph described by an adjacency matrix. We then use this implementation to perform several experiments to study the characteristics outlined in this chapter.

Chapter 4

Performance Analysis

4.1 Introduction

This chapter presents the description of an implementation of the protocol model described in the previous chapter, followed by some experiments to analyse its performance.

Section 4.2 describes the implementation of the matrix iteration algorithm and how the path diversity of the routing solution was measured.

Section 4.3 presents the topologies used in the experiments as well as some results. Section 4.3.1 presents the results for a first test topology based on a highly hierarchical and regular network graph. In section 4.3.2 the results for a second test topology based on real data from a small segment of the European region of the Internet inter AS network are presented. This topology was built from both CAIDAs database and the European Internet Registry (RIPE) information [2].

4.2 Algorithms

This section presents the matrix iteration algorithm to calculate the routing solution and the algorithm used to measure path diversity in that solution.

4.2.1 Matrix Iteration

The matrix iteration algorithm is quite simple and its theoretical explanation can be found in section 3.5. This section explains how to perform the matrix iteration algorithm for

this particular protocol model. First we introduce the data structures for the algorithm.

We start with the data structures for assigning policies/labels to links, and the two policy operations. The policies are expressed in our implementation as integer values, so we match integer values to the sets S and T both shown in section 3.4 equations 3.3 and 3.1 respectively.

The first operation is for policy selection corresponding to the algebraic $\oplus$ operation, which selects the most preferred given two policies. The preference order for both set S and set T are presented by equations 3.4 and 3.2. However, the preference of T is only used to distinguish when both policies have the S_{L2} label.

The second operation is the path composition operation, which calculates the resulting policy from extending two paths, as showed in 3.5. Using the algebraic $\otimes$ operation for set S (table 3.3) and for set T (table 3.2) we obtain the paths weight (w).

From this we build the adjacency matrix to represent a network topology, i.e. each position of the matrix stores a policy/label which is the path value from an origin-destination pair. We represent the network matrix as $A\,[X]\,[Y]$, where the lines are the origins $[X]$ and columns correspond to each a destination $[Y]$.

Note that, the function that populates each position of the matrix also deals with the reverse policy. In other words, if node 0 has the policy value of U_W to node 1. Node 1 has the policy value of D_W to node 0 (if $A(0,1{=}U_W$ then $A(1,0){=}D_W$).

The function to multiply matrices used for the matrix iteration algorithm is similar to the binary operation used in algebra, the difference being that instead of times it is used the path composition and the sum is replaced by the path selection operation. This was showed in section 3.5. In the Path Selection operation more than one node can be preferred as next hops so each position of the matrix has to have also a list of next hops (in

order to be used for routing table). We can see in 3.6 that for this function we have to use three matrices for this operation. Table 4.1 illustrates the data structure used for each matrix positions.

Source	Destination	Path Policy ($w(P)$)		Set of next Hops H_y
X_i	Y_j	$w(P_y) \in S$	$w(P_y) \in T$	$H_y = \{X_1, X_2, \cdots, X_n\}$

Table 4.1: Adjacency matrix A data cell structure

The input for the matrix iteration algorithm is a topology, which contains all connections between neighbours, and the policies applied in those connections, positioning them in a hierarchy. The topology is given through a text file where each origin destination pair have a policy value associated to the link. This data is loaded into the adjacency matrix A.

The matrix iteration algorithm obtains the routing solution A^*, by left multiplies matrix A recursively (equation 3.6) until condition (equation 3.7) is met.

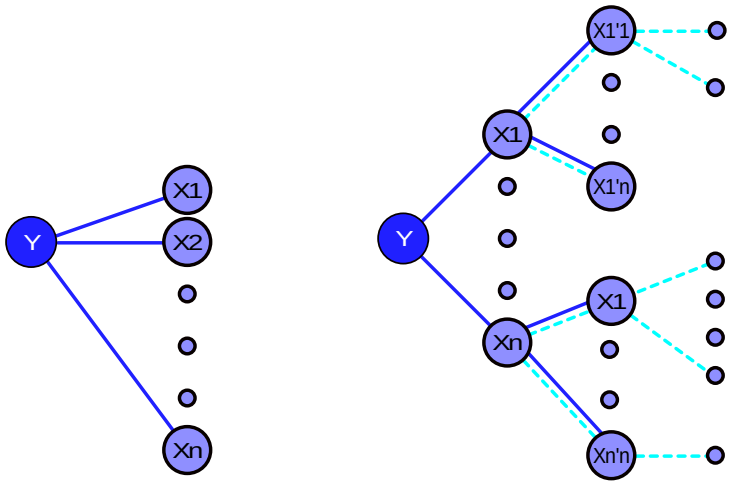
4.2.2 Path Discovery

After calculating a routing solution using matrix iteration algorithm, it is possible to generate the forwarding tables by extracting information from A^*. Note that each line of the matrix corresponds to the forwarding table of one of the nodes. Because matrix A^* contains a set of next hops to each reachable destination, it is possible to identify the existing paths between each pair of nodes.

Path Discovery algorithm explores all forwarding possibilities (for each source-destination pair), that packet can follow hop-by-hop, starting from the a destination and build all the possible paths to each source. If we take a column from matrix A^*, we get the next hops from all sources to a destination. To calculate all paths from a destination to all sources, first we simplify the search by organising the information. Instead of having a source with a list of next hops like a forwarding table we change it to a next hop with a list of sources, i.e. a list of sources that possess the index node as next hop.

The idea is to first add all directly connected hops to the destination, from simplified

search list. These links are also paths from those hops to the destination, as illustrated in figure 4.1(a). Now by adding to this hops their next hops, shown in figure 4.1(b), we can get both paths of sources at two hops distance from the destination and/or alternative paths of sources at one hop distance. By appending hops recursively we get all paths of all sources to the destination. This is only possible if the network is loop free otherwise the algorithm enters a endless loop calculation. We then run the Path Discovery algorithm on every destination, to find all topology paths.



(a) First Hops (b) Second and Third Hops

Figure 4.1: A scheme representing the Path Discovery algorithm

After all possible paths between each pair of nodes in the topology have been found, they can be classified into two categories:

- Disjoint paths - paths that do not share any link or node other than the source and destination;
- Different paths - paths that share at least one link or node but with at least one different.

It is important to distinguish the multiple routes since the goal is to have as many disjointed paths as possible. In figure 4.1 it can be easily seen that the maximum number of disjoint paths is limited by the number of first hops the destination has, therefore we may get the most out of the protocol in a full meshed network.

The forwarding in the protocol is destination-based and hop-by-hop, the actual path followed by packets will depend on how each node distributes traffic between the possible next hops it has in the forwarding table. Take the example of figure 4.2.

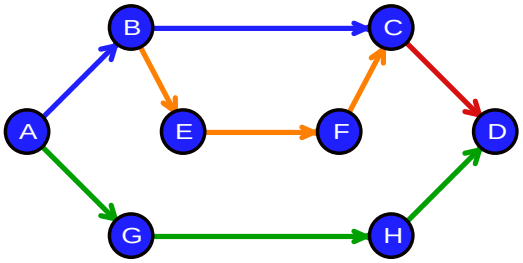


Figure 4.2: Disjoint paths.

The example shows three different paths from A to D and a broken link between nodes C and D. Routing through node B fails since the different path rejoins before the broken link. This simple example shows that in case of a link failure different paths do not guarantee connection. However, this is not true in the disjoint paths case (node G in the example).

4.3 Simulation Results

In order to assess the performance of a protocol with the described model, we performed a set of experiments in different network topologies that we describe in the next subsections.

4.3.1 Network Topology Characteristics

The first topology is a highly regular and hierarchical topology composed of an accesses level, aggregation level and a core level. It is a common network topology design that a Data Centre, University Campus or Enterprise Campus could use. The hierarchy with the redundant connections between levels achieves: robustness, high availability and throughput. Figure 4.3 illustrates the network topology.

We can see that the topology contains 52 nodes and 110 links divided in three hierarchical levels. Blue, green and orange colours represent levels 0 (access), 1 (aggregation) and 2 (core) respectively. We have 36 nodes at level 0, each one of them connected to two nodes at level 1 so that we have at least two paths towards level 1 in those nodes. At level 1 there are 4 clusters of aggregation formed by 3 nodes each, making the total of 12 nodes at level 1.

Links between nodes in different levels are labelled with U_W or D_W labels depending

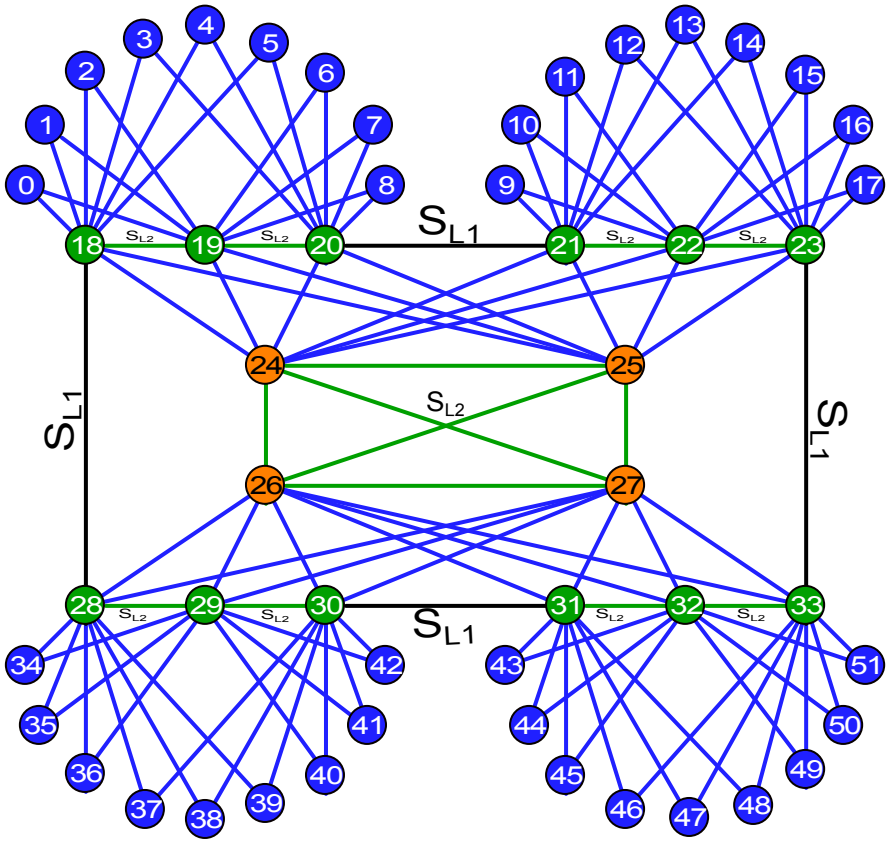


Figure 4.3: Network Topology.

on the direction. Same level link labels are used between nodes in the same level of the hierarchy.

Within a cluster the nodes are connected via S_{L2} links so that local horizontal traffic is possible. S_{L1} links are used between the 4 clusters meaning that horizontal traffic should only occur within the cluster or at most to the neighbouring clusters. Finally each level 1 node is connected to two level 2 nodes via U_W/D_W links.

There are 4 nodes at level 2 that serve as the core (or a local clique for a greater network). They are connected via S_{L2} links. Due to this structure, the highest path diversity expected in the topology is when a node at level 0 tries to reach another level 0 node that is not in its neighbour cluster. Making 2 paths (from level 0 to level 1) x 2 paths (from level 1 to level 2) x 3 paths (within level 2) x 2 paths (from level 2 to level 1) = 24 different paths with at least one different node between them.

Figure 4.4 only shows the same level links in the topology and serves as an example to see how the labelling of S_{L2} links has to be done. Green arrows represent (S_{L2}, L) links and blue lines S_{L1} links . Considering the level 2 nodes several cycles may form.

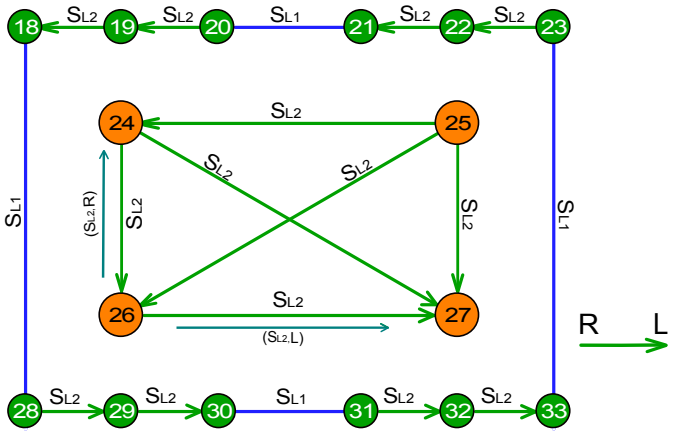


Figure 4.4: S_{L2} Labelling.

The first one is (27, 26, 24, 25, 27) going around this cycle in that direction has at least one differently labelled link due to the link 25-27 being (S_{L2}, R) and all the others (S_{L2}, L). If we circle in the other direction, then all links are (S_{L2}, R) but 25-27 is (S_{L2}, L) for this direction.

Let us now see the crossover links. Consider for example cycle (24, 27, 25, 24). If 24-27 was labelled (S_{L2}, L) then (24, 27, 25, 24) would be free but (24, 27, 26, 24) would be not free since all links would be (S_{L2}, L). The solution is to label 24-27 (S_{L2}, R). (24, 27, 25, 24) is still free and (24, 26, 27, 24) is now also free. The same reasoning is behind the labelling of link 26-25 as (S_{L2}, L). So if in the figure 4.4 a closed path of green arrows was present, it would mean that a cycle existed.

4.3.1.1 Network Topology Experiments

We calculated the routing solution for the topology with the described labelling of the links and then measured the amount of different paths for all destination-source pairs of the topology. We obtained three forwarding entries for 1.96% of the source-destination pairs, two forwarding entries for 89.9% and one forwarding entry for 8.14%. The topology

is designed so that for almost every destination, each node has at least two forwarding options, and it was to be expected that the majority of the pairs had two different entries in the forwarding table. The 8.14% of pairs with a single entry are related with three situations:

(a) Level 0 nodes that are reachable from level 1 nodes via the single D_W link which is more preferred than the alternative paths via S_{L2} links through the neighbours at the same level;

(b) S_{L1} links amongst the four level 1 aggregation clusters. These links create single entries of weight S_{L1} amongst the neighbouring nodes of different clusters and these are more preferred than the paths available via the higher level nodes. This preference for proximity between clusters reduces path diversity since going via higher levels is less preferred, but it is a design choice. In any case, these higher level links (as the S_{L2} in situation (a)) are available in terms of failures;

(c) the labelling of S_{L2} in T (sub-labels L and R) at level 2 nodes causes some of the level 2 nodes to have just one path to its neighbours to avoid cycles with only right (or left) links. However, as a trade-off there are some nodes of level 2 with three entries.

The existent paths between each pair of nodes were calculated using matrix A^*. We then classified them into disjoint paths and different paths. The graph of figure 4.5 shows the cumulative distribution for the number of paths per source-destination pair considering both disjoint and different paths.

The ideal curve would remain low, and increase at the end (meaning that a large percentage of nodes have high path diversity). In the topology every node has at most three links with nodes at the same label, meaning that, at most there are three disjoint paths between any pair of nodes. We can see in the figure 4.5 that three is in fact the limit in the number of disjoint paths. However, we see that 13.1% of the source destination pairs only have one disjoint path, 86.7% have two and 0.2% have three disjoint paths between them. Considering the different paths we see that only 8.1% of the pairs have a single path. This is related with the three cases above where only one entry exists in the forwarding

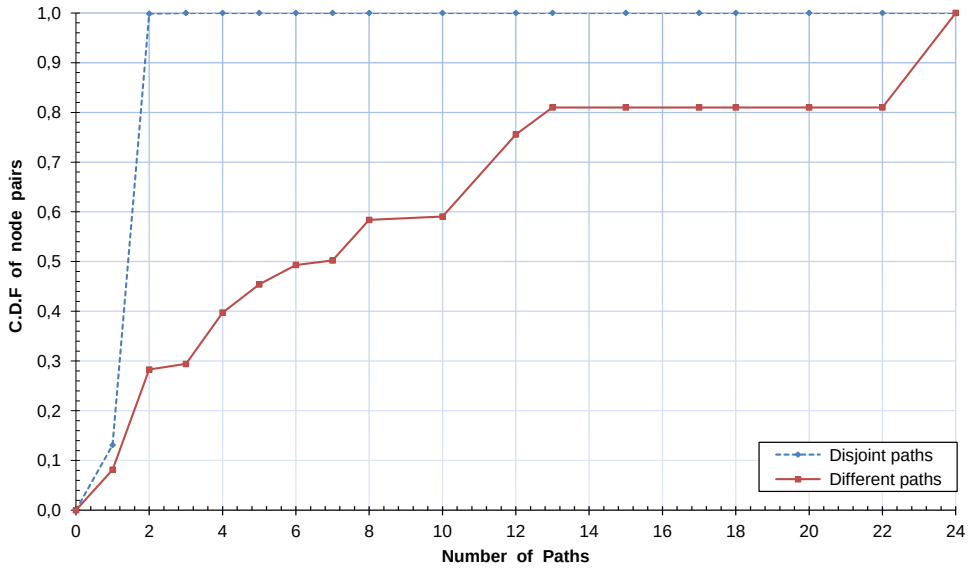


Figure 4.5: Cumulative distribute function of the number of different paths per source destination pairs.

table. They are due to the design choices at certain parts of the network that prefer the closest path to the destination instead of the direction that provides more links. The path diversity is higher in this case, with the highest number of different paths between some pairs reaching the value of 24. As seen in the figure 4.5, 29% of the pairs have only three or less paths, 50% have seven or more possible paths, and 19% have 13 or more different paths. This simple example of a common hierarchical network illustrates the path diversity that can be used simultaneously when using such a protocol. This maintains the simplicity of simple destination based hop-by-hop forwarding. Making use of a careful labelling of the links it is possible to use all links and make use of the path diversity introduced in redundant hierarchical network topologies.

4.3.2 Internet Topology Characteristics

The Internet is a completely different type of network. It has a much more loose hierarchy. A certain hierarchy is induced by the business relationships between the ASes, but in this case there is not a clean structure with clear levels like in the previous topology. There are links directly connecting lower level nodes with higher level nodes (without passing through intermediate levels) and also same level links between different levels of the hierarchy. In order to understand the model behaviour in such a network we used a

topology of a small segment of the Internet network at the inter-AS level. The topology was obtained from CAIDAs AS Relationships Data Research project and cross matched with information from the European Internet registry (RIPE) in a previous work [2]. The topology has 54 ASes, a bigger topology could have been used however excessively computer resources would be consumed. The association between the nodes number and the real AS numbers is not done because it is irrelevant to our experiments.

For this topology we defined a loose hierarchy mapping the usual Internet business relationships of "Provider-Customer" and "peer" to our U_W/D_W and S_L policies. The top nodes (orange nodes) in figure 4.6 are the highest-level nodes (nodes that have no U_W links) and the figure is drawn such that as we descend in the figure the level degree also follows (nodes with progressively more U_W links as we come down in the Figure). The purple nodes in the bottom are the lowest-level nodes (nodes with no D_W links).

The Internet inter AS network is closely modelled by a long-tail distribution of node degree. This means that a small number of nodes have high connectivity and a large number of ASes has a small number of connections. Our topology adapted from previous studies [2] corresponds to the part of the network that is densely connected and the long-tail of smaller ASes is left out. The only smaller, less connected ASes present are the ASes in the Portuguese region. Therefore, the topology in figure 4.6 is a massively connected network. The connections are distinguished by same level policy links (S_L) with the blue (cyan) connections and downwards (D_W) and upwards (U_W) policies links represented in purple. It is notorious the predominance of same level connections in this topology and the vast majority are found between nodes at higher levels.

The graph of figure 4.7 shows the cumulative distribution function (C.D.F) for the number of connections per node (node degree) considering same level links and different level links. We can see in the figure 4.7 that the C.D.F of the number of nodes grows slowly for same level links meaning that those are more present in the topology than different level links. It can be observed that 50% of the topology nodes have more than 13 same level links and

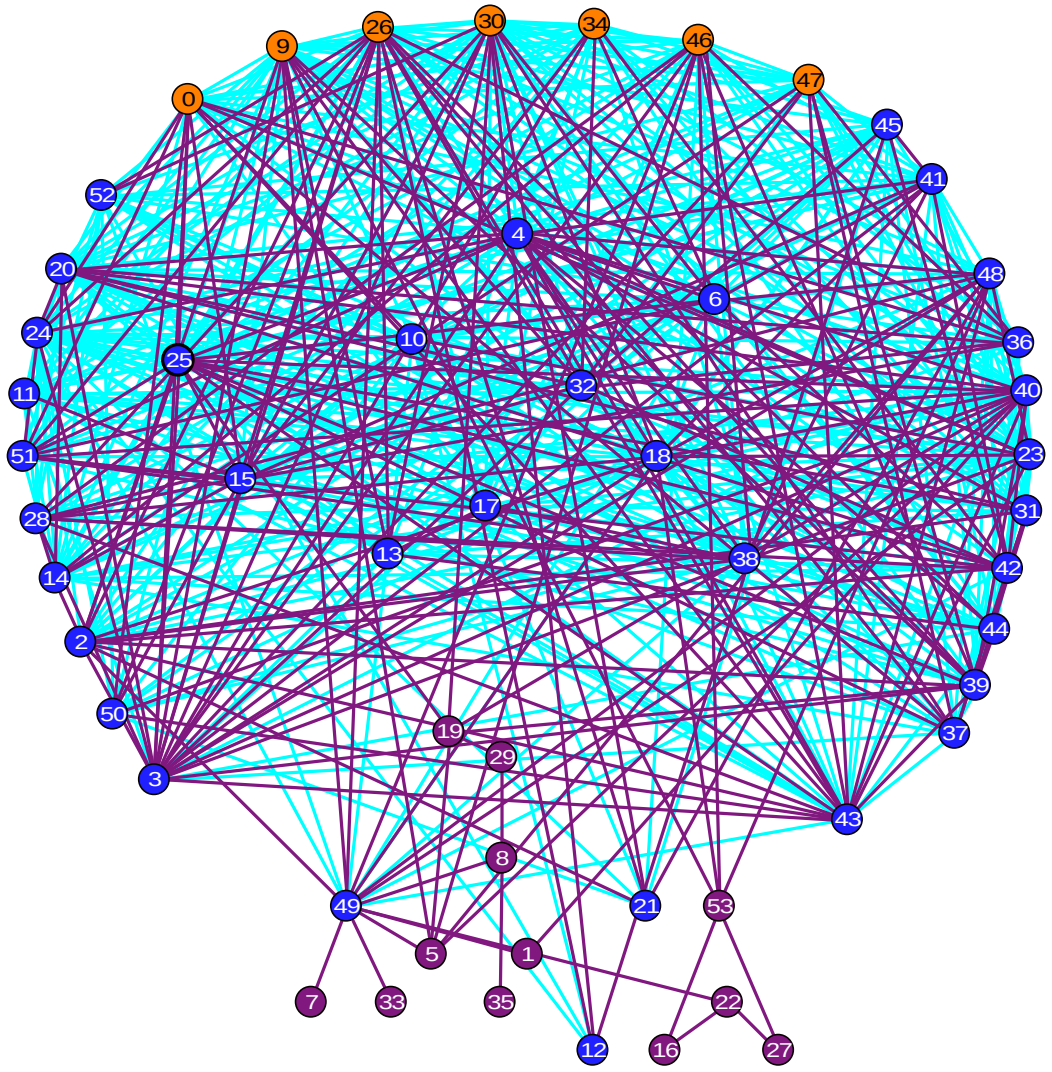


Figure 4.6: CAIDA Topology.

only 40% have 8 different level links.

4.3.2.1 Internet Topology Experiments

We ran three experiments with the topology. On the first experiment, all same level links were labelled with S_{L1} policy. On the second and third experiments some of the same level labels were modified from S_{L1} policies to S_{L2} policies.

In the second experiment, S_{L1} policies were changed in the nodes with a great number of same level connections; eight nodes were selected based on this criteria (9, 11, 17, 20, 23, 24, 30 and 45). In the third experiment we changed the highest-level nodes

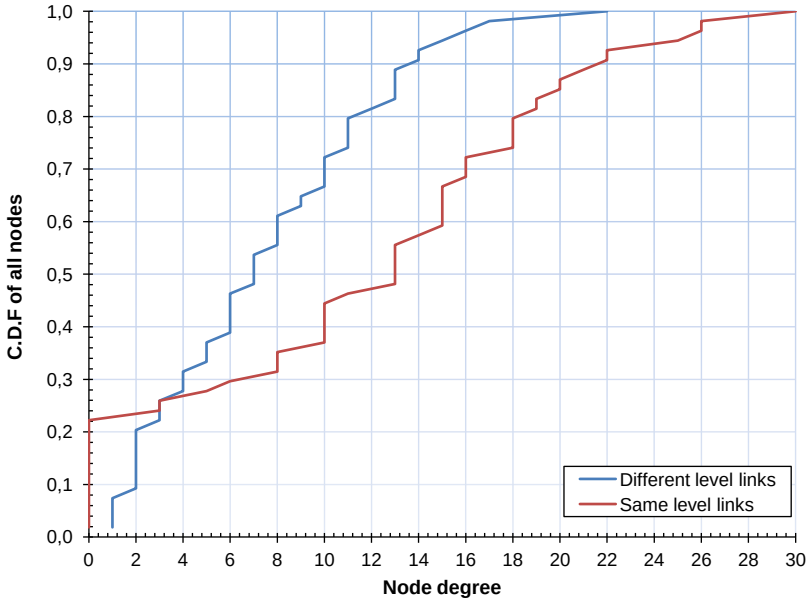


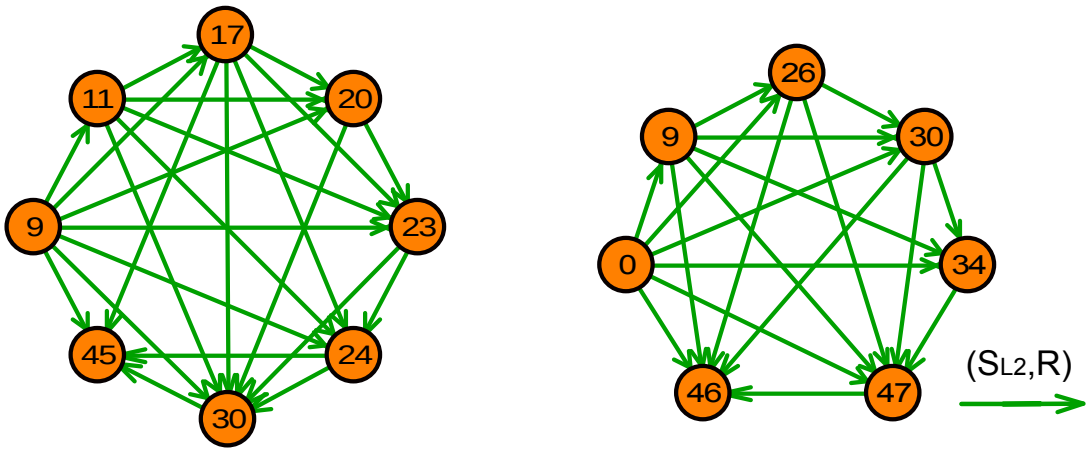
Figure 4.7: CAIDA Links Distribution.

(the nodes with no U_W links); seven nodes were selected (0, 9, 26, 30, 34, 46 and 47). We changed the same level links between these nodes (the links connecting them directly) since the impact of changing these labels should affect more the number of paths in the topology.

For a better perception of the results we will compare the second and third experiment results against the first experiment to see the impact of these changes. Figure 4.8 shows how the S_{L2} labelling that was done for the two last experiments. As can be seen in both cases the cycles are free. We calculated the routing solution and classified the different paths between each pair of nodes in to different and disjoint paths as we did in the experiments of the previous topology.

The graph of figure 4.9 considers all nodes in the topology and shows the cumulative distribution for the number of different paths per source-destination pairs in the first and second experiments.

The distribution was divided in two graphics, because as we can see there are 80% of nodes pairs in the topology with less than 60 paths and the other 20% have more than 1000 paths. So, to scale the graphs we divided them. We can see a slight improvement in the number of different paths each node has for the second experiment. In fact table 4.2 shows that number of different paths increased 8.11% compared to the first experiment.



(a) Second Experiment (b) Third Experiment

Figure 4.8: S_{L2} Labelling

Path	1°Exp.	2°Exp.	[%]
Different	157215	169960	8, 11
Disjoints	6362	6334	−0, 44

Table 4.2: Path diversity values between first and second experiments

All nodes increased the total number of paths to all destinations, but the highest increase was for node 3, which increased the path diversity from 9606 to 10168. One possible reason for this increase is that nodes from high levels gain paths between them by being able to use paths with multiple consecutive same level links. However, for the lower level ones, the beginning of the paths (going up in the hierarchy) remains the same and the diversity only occurs in the parts of the paths that traverse the higher level nodes. This is the reason why the number of different paths increases while the number of disjoint paths remains almost the same.Another reason related to the previous one is that some of selected nodes are not directly connected (visible in figure 4.8(a)), meaning that they are either connected by paths with nodes at higher-level or do not have paths connecting them at all.

Figure 4.10 illustrates the improvement of the path diversity from all nodes to the modified nodes, distinguishing the paths in different and disjoint paths. We can clearly see that the

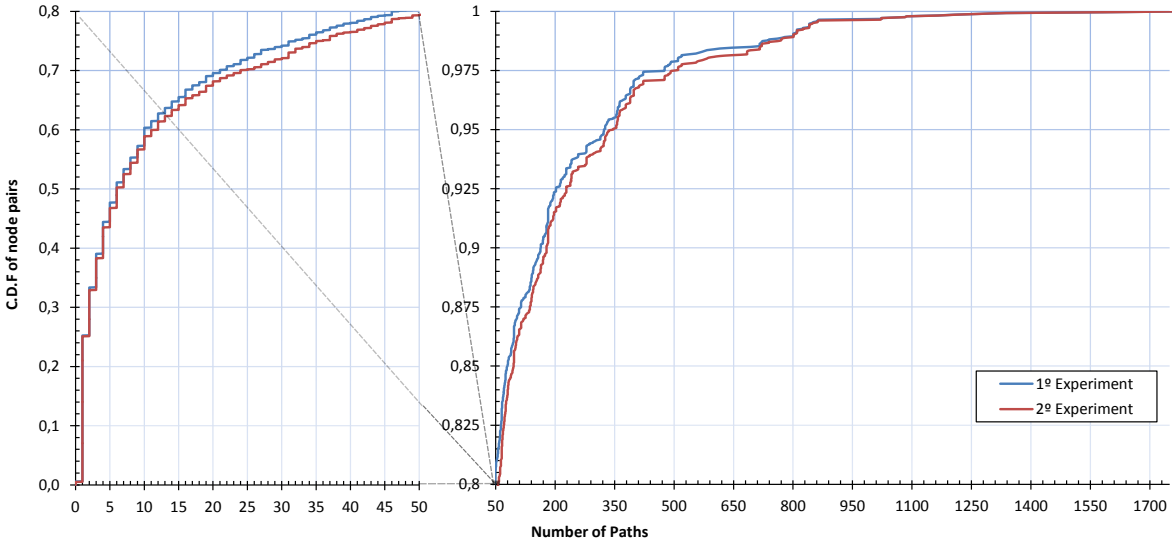
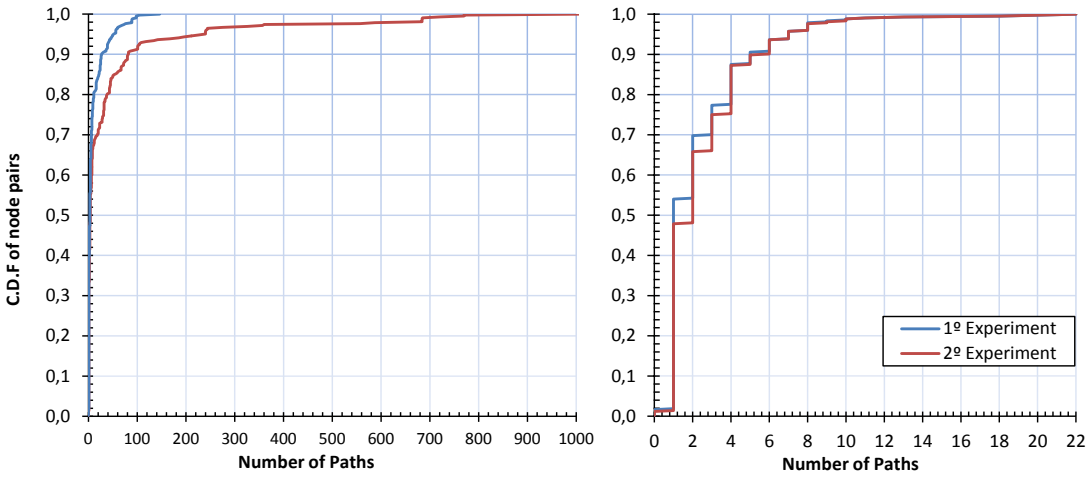


Figure 4.9: The number of different paths cumulative distribute function per source destination pairs.

biggest difference is indeed for the number of different paths, as shown in figure 4.10(a). 10% of the nodes in the first experiment have more than 27 different paths and the maximum of different paths is 147. As for the second experiment, 10% of the nodes have more than 81 paths and the maximum number of different paths is 1020. Despite the loss of 0.44% disjoint paths in the topology overall, there is a 6% (61 more paths) increase illustrated in figure 4.10(b) of the number of disjoint paths from all nodes to the selected nodes.



(a) Different paths difference

(b) Disjoint paths difference

Figure 4.10: Results of the 1º and 2º experiment, path diversity to the selected nodes.

Path	1°Exp.	3°Exp.	[%]
Different	157215	233730	48, 67
Disjoints	6362	6368	0, 09

Table 4.3: Path diversity values between first and third experiments

In the previous results, the highest-level nodes benefit the most of using S_{L2} links, so in the third experiment as it was mention in subsection 4.3.2.1 we changed policies in this top-level nodes (nodes without U_W links). We can see the results on figure 4.6 in orange.

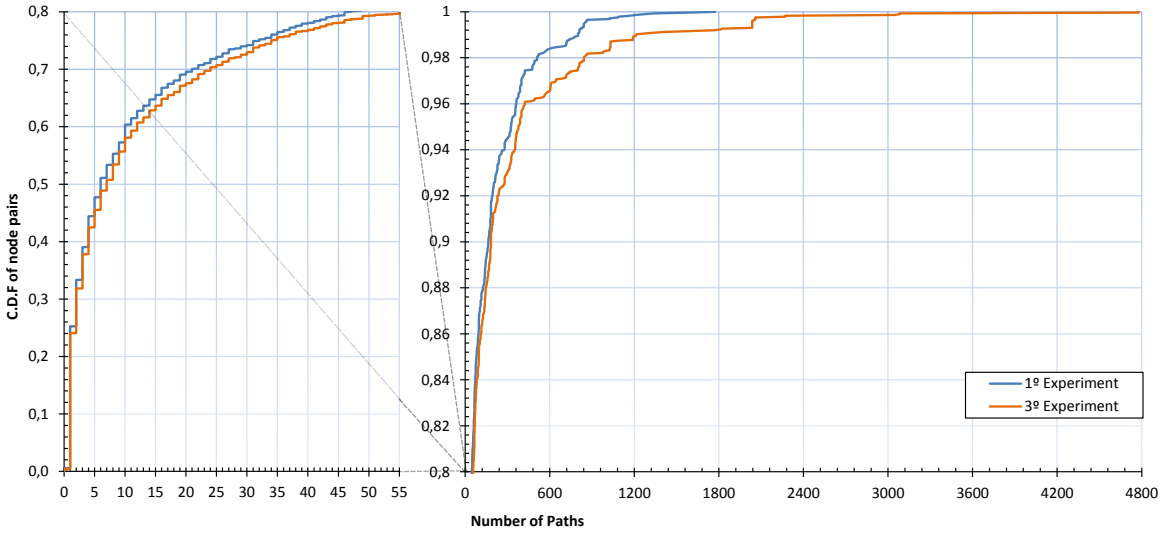
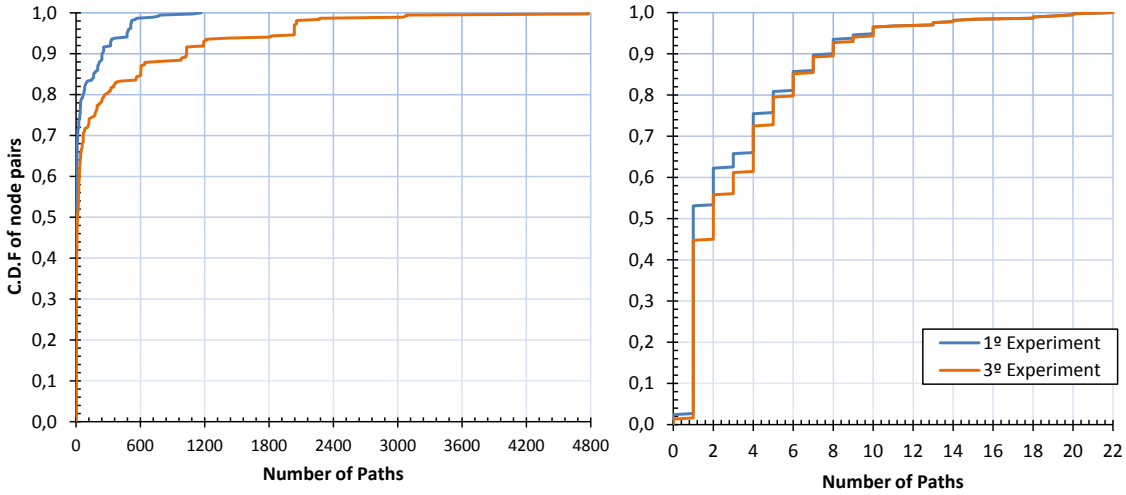


Figure 4.11: Cumulative distribute function of the number of different paths per source destination pairs.

Comparing with the first experiment results we can notice a notorious improvement in the number of different paths in third experiment; table 4.3 shows the results.

In figure 4.11 it can be seen that the number of paths increases significantly for 4% of the node pairs. Some node pairs can now have 4778 paths, which is also the maximum of path diversity. In terms of the overall disjoint paths there is a 0.09% improvement, representing an increase of 6 paths.

By just changing some few policies in some selected nodes, we were able to increase the number of different paths in almost 50%. The much lower increase in the number of disjoint paths is due to the lack of alternative paths in the lower nodes for the top nodes. So, establishing paths between higher level nodes only improves disjoint paths between them, but still adds a significant path diversity overall. For instance, in the first experiment



(a) Different path difference

(b) Disjoint path difference

Figure 4.12: Difference in between the first and third experiments

node 34 did not have paths to nodes 26 and 46 because it was not directly connected (seen in figure 4.8(b)). However, node 34 in third experiment was able to established a path to them, affecting the path diversity on these nodes and in the overall topology.

Remember that, the ideal curve would remain low and increase at the end, meaning that a large percentage of nodes have high path diversity. In both cases of figure 4.12 the improvement is very clear from the first experiment to the third. Consider figure 4.12(a), 90% of nodes pairs in the first experiments have 244 or less paths and for third experiment have 1031 or less. The difference between them rises for the remaining 10%. In figure 4.12(b) the number of disjoint paths increased by 10%, which means, there are more 101 disjoint paths to the modified nodes (total of 1281 disjoint paths).

In summary, the addition of a secondary algebra with the lexicographic product to allow the use of paths with multiple links in the same level proved to be useful in networks with a power law distribution such as the Internet. As proven by the experiments, in the more densely connected areas path diversity increased and in the majority of cases totally disjoint paths also increased. For lower levels nodes on less connected areas the path diversity between them was maintained. However, if some paths between them go through the dense area (going upward in hierarchy) the path diversety increases.

Chapter 5

Conclusions

Multipath routing using policies is more complex than traditional single shortest path protocols and the conditions in which it correctly operates are more difficult to understand and define. In [1] an algebraic model is used to obtain and systematize the convergence conditions for these types of protocols. In this thesis we used this model and designed a multipath policy routing protocol that uses simple destination-based hop-by-hop forwarding. The choice to use a *non decreasing* model, to increase the number of paths that can be considered equally good (and therefore be used simultaneously) implies that the network has to be restricted in order to assure correct behaviour. Two problems can occur in such a model if the network is not restricted: the multipath routing solution can be infinite and forward loops can occur. Both problems can be modelled by the existence of a *non free* cycle.

A trade-off was clearly identified between the number of policies that maintain preference and the existence of more *non free* cycles in the network. More policies that maintain preference with the addition of links mean more equally preferred paths and more multipath but they lead to more cycles being problematic and therefore forbidden. A compromise has to be found and some path compositions need to be considered invalid in order to get a good balance between the level of multipath and the restrictions needed on the network. Another identified limit is the use of policies that are symmetrically applied in the links. When applied to more than two consecutive links in a cycle those policies cause forwarding

loops, and if such a semantic is needed then a solution like the use of a secondary algebra on those links has to be used.

We presented a generic multipath policy-based hierarchical routing protocol to be used in networks with distributed control and independent nodes, that converges to a solution in finite time and does not have forwarding loops. Our protocol resolves the symmetrical policy problem paths within the same level of the hierarchy using a lexicographic product with a second algebraic set to allow loop *free*.

In summary, we offered the possibility to have paths with different number of hops to be considered of equal preference (*non decreasing* $\otimes$ operation) and therefore some paths needed to be invalid (otherwise the network would have to be acyclic).

The measured path diversity in the experiments with hierarchical topologies showed how such a simple protocol can make use of the path diversity of a network and perform multipath policy routing. It also highlighted that there are limits for such an approach and that careful application of the policies has to be done. Overall, this work provides an insight on the fundamental limits and trade-off's involved in achieving such results for these types of protocols. The practical implications of implementing a multipath policy-based routing protocol go beyond the models used to prove that they operate correctly. The conclusions in our work show how within the restrictions for correct operations different amounts of traffic diversity and network restrictions can be achieved depending on some decisions on the policies to apply and where to apply them.

5.1 Future work

During the development of the proposed solution, the network topologies were designed in paper and then converted to a text file format (.txt), but network restrictions (cycles) had to be verified beforehand. Taking this into account, it would be interesting in future work to use the model to devise a tool that verifies correct operation and checks the network for *non free* cycles. The model can also be used to suggest topology designs and link policy assignments according to degrees of multiple paths wanted between nodes.

The labelling of the graph defines the usable paths and therefore the base on which traffic engineering can be performed. It would be interesting to devise methods that given a set of policies and there preference, finds the most effective labelling of a graph so that a particular objective for the routing solution is found i.e. to have more paths through the most efficient parts of the network or to maximize the average number of equally preferred paths per destination.

Bibliography

[1] Pedro Amaral. *Multipath Inter-domain Policy Routing.* Phd thesis, Universidade Nova de Lisboa, 2012.

[2] Cláudio Assunção. Hybrid link-state path-vector protocol ++, 2012.

[3] John S Baras and George Theodorakopoulos. Path problems in networks. *Synthesis Lectures on Communication Networks*, 3(1):1–77, 2010.

[4] I. Castineyra, N. Chiappa, and M. Steenstrup. The nimrod routing architecture. RFC 1992, August 1996.

[5] Chi-kin Chau. Policy-based routing with non-strict preferences. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 387–398. ACM, 2006.

[6] C. Cheng, R. Riley, S. P. R. Kumar, and J. J. Garcia-Luna-Aceves. *A loop-free extended Bellman-Ford routing protocol without bouncing effect*, volume 19. ACM, New York, NY, USA, September 1989.

[7] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms.* McGraw-Hill Higher Education, 2nd edition, 2001.

[8] Edsger. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[9] J. Doyle and J.D.H. Carroll. *Routing TCP/IP 1.* CCIE Professional CCIE CCIE Professional Development Series. Cisco, Pearson Education, 2005.

[10] Francisco Ganhão. Multi-region routing, 2009.

[11] Igor Ganichev. *Interdomain Multipath Routing.* PhD thesis, University of California, Berkeley, 2011.

[12] P Godfrey, Igor Ganichev, Scott Shenker, and Ion Stoica. Pathlet routing. *ACM SIGCOMM Computer Communication Review*, 39(4):111–122, 2009.

[13] Michel Gondran and Michel Minoux. *Graphs, dioids and semirings: new models and algorithms*, volume 41. Springer, 2008.

[14] Timothy G Griffin. The stratified shortest-paths problem. In *Communication Systems and Networks (COMSNETS), 2010 Second International Conference on*, pages 1–10. IEEE, 2010.

[15] Timothy G Griffin and Alexander JT Gurney. Increasing bisemigroups and algebraic routing. In *Relations and Kleene algebra in computer science*, pages 123–137. Springer, 2008.

[16] Timothy G Griffin, F Bruce Shepherd, and Gordon Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking (ToN)*, 10(2):232–243, 2002.

[17] Alexander James Telford Gurney. *Construction and verification of routing algebras.* PhD thesis, PhD thesis, University of Cambridge, 2009.

[18] Alexander JT Gurney and Timothy G Griffin. Lexicographic products in metarouting. In *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, pages 113–122. IEEE, 2007.

[19] C. Hopps. Analysis of an Equal-Cost Multi-Path Algorithm. RFC 2992 (Informational), November 2000.

[20] Aaron D Jaggard and Vijay Ramachandran. Relating two formal models of path-vector routing. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 619–630. IEEE, 2005.

[21] Nate Kushman, Srikanth Kandula, Dina Katabi, and Bruce M Maggs. R-bgp: Staying connected in a connected world. USENIX, 2007.

[22] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. Delayed internet routing convergence. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '00, pages 175–187, New York, NY, USA, 2000. ACM.

[23] G Lee and J Choi. A survey of multipath routing for traffic engineering. *Information and Communications University, Korea*, 2002.

[24] Yong Liao, Lixin Gao, Roch Guerin, and Zhi-Li Zhang. Reliable interdomain routing through multiple complementary routing processes. In *Proceedings of the 2008 ACM CoNEXT conference*, page 68. ACM, 2008.

[25] G. Malkin. RIP Version 2. RFC 2453 (Standard), November 1998. Updated by RFC 4822.

[26] Murtaza Motiwala, Megan Elmore, Nick Feamster, and Santosh Vempala. Path splicing. *ACM SIGCOMM Computer Communication Review*, 38(4):27–38, 2008.

[27] John Moy. Ospf version 2. Internet RFC 2328, April 1998.

[28] Paolo Narvaez, K-Y Siu, and H-Y Tzeng. Efficient algorithms for multi-path link-state routing. ISCOM'99, November 1999.

[29] D. Oran. OSI IS-IS Intra-domain Routing Protocol. RFC 1142 (Informational), February 1990.

[30] Byrav Ramamurthy, George Rouskas, and Krishna Sivalingam(eds.). *Next-Generation Internet Architectures and Protocols*. Cambridge University Press, 2011. (under preparation).

[31] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006.

[32] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), January 2001.

[33] John Scudder, Alvaro Retana, Daniel Walton, and Enke Chen. Advertisement of multiple paths in bgp. 2012.

[34] J.L. Sobrinho. An algebraic theory of dynamic network routing. *Networking, IEEE/ACM Transactions on*, 13(5):1160–1173, 2005.

[35] João Luís Sobrinho. Algebra and algorithms for qos path computation and hop-by-hop routing in the internet. *IEEE/ACM Transactions on Networking (TON)*, 10(4):541–550, 2002.

[36] João Luís Sobrinho, Timothy G Griffin, and Michaelmas Term. Routing in equilibrium. *Mathematical Theory of Networks and System*, 2010.

[37] Andrew S. Tanenbaum and David J. Wetherall. *Computer Networks*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition, 2010.

[38] Srinivas Vutukury and JJ Garcia-Luna-Aceves. *A simple approximation to minimum-delay routing*, volume 29. ACM, 1999.

[39] Wen Xu and Jennifer Rexford. *MIRO: multi-path interdomain routing*, volume 36. ACM, 2006.

[40] Xiaowei Yang, David Clark, and Arthur W Berger. Nira: a new inter-domain routing architecture. *Networking, IEEE/ACM Transactions on*, 15(4):775–788, 2007.

[41] W. Zaumen and J. J. Garcia-Luna-Aceves. Dynamics of link-state and loop-free distance-vector routing algorithms, 1992.