Interdomain Traffic Engineering with MPLS

Cristel Pelsser

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

October 24, 2006

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

Interdomain Traffic Engineering with MPLS
by Cristel Pelsser

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

This work was supported by the European Commission within the ATRIUM project and by the Walloon Government under the WIST TOTEM project. This work was also partially supported by the European Union under the E-Next project FP6-506869.

Contents

List of Figures

List of Tables

Acronyms

ABR Area Border Router

Adj-RIB-In Adjacency Routing Information Base Incoming routes

AS Autonomous System

ASBR AS Border Router

ASN AS Number

ATM Asynchronous Transfer Mode

ATRIUM A Testbed of Terabit IP Routers running MPLS over DWDM

BGP Border Gateway Protocol

BRPC Backward Recursive PCE-based Computation

CPS Confidential Path Segment

CSPF Constrained Sortest Path First

DAMOTE Decentralized Agent for MPLS Online Traffic Engineering

DiffServ Differentiated Services

DNS Domain Name Server

DPC Distributed Path Computation

eBGP external BGP

ECMP Equal Cost Multi-Path

ERO Explicit Route Object

EXRS eXclude Route Sub-Object

Gbps Gigabits per second

GMPLS Generalized Multi-Protocol Label Switching

HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
iBGP	internal BGP
ICMP	Internet Control Message Protocol
ID	Identifier
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IntServ	Integrated Services
IP	InternetProtocol
IPDA	IP Destination Address
IS-IS	Intermediate-System to Intermediate-System
ISP	Internet Service Provider
IX	Internet eXchange
LDP	Label Distribution Protocol
Loc-RIB	Local Routing Information Base
LSP	Label Switched Path
LSR	Label Switching Router
Mbps	Megabits per second
MED	Multi-Exit Discriminator
MP	Merge Point
MPLS	Multi-Protocol Label Switching
NH	Next-Hop
NNHOP	Next-Next-Hop
OPCA	Overlay Policy Control Architecture
OSPF	Open Shortest Path First
POP	Point of Presence

PCC Path Computation Client

PCE Path Computation Element

PCErr Path Computation Error

PCRep Path Computation Reply

PCReq Path Computation Request

PE Provider Edge

PKS Path Key Sub-object

PLR Point of Local Repair

PML Path Merge LSR

PSL Path Switch LSR

QoS Quality of Service

RON Resilient Overlay Network

RR Route-Reflector

RRO Record Route Object

RSVP Resource reSerVation Protocol

RTT Round Trip Time

SDH Synchronous Digital Hierarchy

SLS Service Level Specification

SONET Synchronous Optical NETwork

SP Service Provider

SPF Shortest Path First

SRLG Shared Risk Link Group

TCP Transmission Control Protocol

TE Traffic Engineering

TED Traffic Engineering Database

TOTEM TOolbox for Traffic Engineering Methods

UcL Unversité catholique de Louvain

VoIP	Voice over IP
VPN	Virtual Private Network
XRO	eXclude Route Object

Chapter 1

Introduction

The Internet initially connected a few tens of routers. It was designed to provide a best-effort service. It was not created for the provision of Quality of Service (QoS) but for the provision of connectivity between the nodes. In the evolution of the Internet from a research to a commercial network, Service Providers (SPs) appeared. Service Providers earn money by providing connectivity to the Internet and/or a transit service to client SPs. After the emergence of SPs came the need for providing more elaborated services than best-effort. Two architectures have been proposed to allow the Internet to support other types of services. The Integrated Services architecture (IntServ) [BCS94] aims at providing stringent end-to-end guarantees to individual flows. The Differentiated Services architecture (DiffServ) [BBC+98] aims at providing several grades of service to different aggregated flows.

Later, the notion of Internet Traffic Engineering (TE) was introduced [ACE+02]. Internet TE is the set of techniques that enable to better control the flow of packets inside a network in order to achieve performance objectives. An overview of these techniques and the principles for Internet TE is provided in [ACE+02]. Internet TE implies the measurement, modeling, characterization, and control of Internet traffic [AMA+99].

Awduche et al. [AMA+99] show that the "major goal of Internet Traffic Engineering is to facilitate efficient and reliable network operations while simultaneously optimizing network resource utilization and traffic performance". From this statement we highlight three objectives:

1. The optimization of the utilization of the resources by avoiding congestion when some resources remain underutilized and thus by increasing the amount of traffic that can be carried by the network.
2. The provision of Quality of Service (QoS) to the end-user flows.
3. The robustness of the network in case of failures.

Thus, now the objective is not only to provide a good service to the users but also to manage networks in a efficient way. This enables Service Providers (SPs) to increase their benefits by supporting a larger amount of traffic, satisfying a larger number of users, satisfying more demanding users and providing competitive service quality with low service disruption.

The Internet is composed of an interconnection of SP networks. SP networks comprise one or a few Autonomous Systems (ASs) that are connected to the rest of the Internet. Among the intradomain routing protocols, OSPF and IS-IS are link-state routing protocols. With OSPF and IS-IS, all the nodes inside the AS learn the complete map of the AS, when the AS is not divided into areas.

An AS is a set of network elements administrated by the same company and that implement consistent routing policies. Subramanian et al. [SARK02] say that there are mostly two types of relationships between neighboring ASs: client-provider and peer-peer peering relationships. In the client-provider relationship, the client pays for Internet connectivity to its providers. One commonly says that the provider provides transit service to its clients. The two ASs that participate in a peer-peer relationship share the cost of this interconnection. Each peer provides a limited transit service to its peer. It accepts to only carry the traffic of the peer and the peer's clients. These relationships are translated into routing policies that constrain the routing information that can be advertised to neighboring ASs. The routing protocol that implements these policies is the Border Gateway protocol (BGP) [Ste99]. BGP is currently used to distribute reachability information between more than 20000 ASs [Hus06b]. The routing information distributed by BGP is limited in order for it to scale. A router is not able to build a complete map of the Internet from the information that is distributed with BGP. Thus, the routing protocols that are used inside ASs are different from BGP.

In the following sections, we introduce the TE techniques used to control the flow of the traffic inside an Autonomous System. We distinguish techniques based on pure IP forwarding from techniques made possible by Multi-Protocol Label Switching (MPLS). We present the characteristics of these techniques.

1.1 Intradomain TE Techniques

Inside a single AS, several techniques can be used to control the flow of the IP packets. They can be divided in two classes : the techniques usable in pure IP networks and those that rely on the use of Multi-Protocol Label Switching (MPLS) [DR00].

1.1.1 IP-based Solutions

The IP packets are routed based on the IP address of the destination. At each hop, the router determines on which interface to forward the IP packet based on the destination stored in the packet header and its forwarding table. At a router, all the packets with the same destination are sent on the same interface and they follow the same path to the destination, without Equal Cost Multi-Path (ECMP).

In a pure IP network, the flow of the IP packets can be controlled by tuning the intradomain routing protocol (also called Interior Gateway Protocol - IGP). Inside a domain, the routers compute the best path to reach each destination in order to populate their forwarding table. The best path is usually the path with the smallest cost where the cost of a path is the sum of the cost of all the links that compose the path. The cost of the links is distributed by the routing protocol. The cost associated with each link depends on the links or paths that the Internet Service Provider (ISP) wants to favor. If each link has a unitary cost, then the routing protocol favors paths with the smallest number of hops. If the cost of each link is a function of the link transmission delay, the routing protocol selects paths with the shortest delay. If the cost associated with a link is a function of the link bandwidth, the routing protocol favors high bandwidth paths.

For example, in figure 1.1, if all links have a unitary cost, the shortest path from the source node S_1 to the destination node D_1 is the path $(S_1, R_{18}, R_{16}, R_{15}, D_1)$ having a total cost of 4. The path used between source S_2 and destination D_2 is $(S_2, R_{11}, R_{18}, R_{16}, R_{15}, D_2)$. If there is a lot of traffic from $S1$ to $D1$ and from $S2$ to $D2$, then the $R_{18} - R_{16}$ and $R_{16} - R_{15}$ links might become the bottleneck since these two links are used by both flows. A common method to redirect traffic away from congested links is to tune the cost of key links. In figure 1.1 it is possible to force the traffic flow $S_2 - D_2$ (resp. $S_1 - D_1$) to follow the path $R_{11} - R_{12} - R_{13} - R_{14} - R_{15}$ (resp. $R_{18} - R_{19} - R_{17} - R_{15}$) by using a cost of 2 instead of 1 on link $R_{11} - R_{18}$ and $R_{16} - R_{19}$ and 3 instead of 1 on link $R_{18} - R_{16}$. In fact, if the traffic demand between each source-destination pair on the network is known, the setting of the link costs can be converted in an optimization problem [FT00, FT04] that can be solved by using appropriate mathematical methods. However, these methods are only useful in practice if the traffic demand is relatively stable. If the traffic demand changes frequently, it will be difficult to dynamically recompute the optimal setting of the link costs and dynamically reconfigure all routers without affecting the current traffic inside the network. There are solutions such as [FT02] to optimize the link costs if the different traffic matrices that may be encountered are known. In addition, a topology change, as a link or node removal for example, may cause links to become congested. Thus, solutions such as [FT03] that aim at finding a set of link costs that is robust to failures are required.

Usually, IP routers select one best path to reach each destination. Depending on the network topology and load, this may cause some links to become congested. When several equal cost paths exist to a given destination, routers can be configured to use all these paths instead of a single one. Several methods exist to distribute the IP packets on these best paths [TH00]. A first solution is to send the packets on each path on a round-robin basis. However, in this case packets from the same TCP connection may be sent on different paths and this may cause packet reordering. Such packet reordering may decrease the throughput achieved by protocols like TCP. Another solution is to ensure that all packets from a single TCP connection are always sent on the same path. In this case, TCP is not be affected, but all paths will not usually carry the same amount of traffic since the volume of TCP

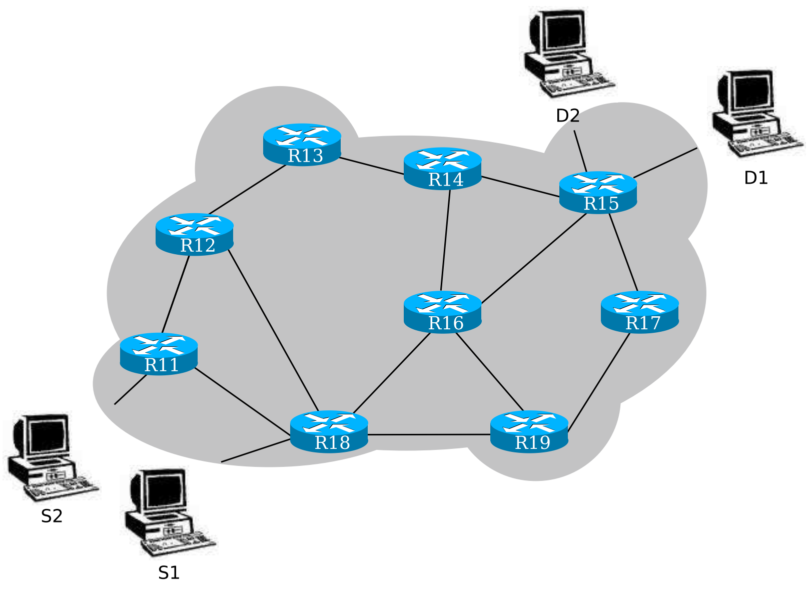


Figure 1.1: A simple ISP network

connections has a heavy-tailed distribution (i.e. there are many TCP connections with a few packets and a few TCP connection with a very large number of packets).

In a pure IP network, providing a good service also implies that the service should not be severely affected by link failures. There are two possible solutions to respond to a link failure. First, the failure can be handled by the lower layers (e.g. ATM or SONET/SDH) and it can be quickly restored in this layer. In this case, the IP network will not be aware of the failure. However, in recently deployed IP networks, IP is used directly over the physical layer and there is no restoration at layer 2. In such a network, a link or router failure must be detected and restored at the IP layer. When a failure occurs, it is detected by the routers attached to the failed resource. The routing protocol used by these routers then floods a message announcing the failed resources to all the routers of the network. Upon reception of this message, each router recomputes its routing table to divert traffic away from the failed resource. The time required to respond to a failure thus depends on the time to flood the failure notification to all routers and the time to recompute the routing table in each router [FFEB05]. Techniques for fast reroute in IP networks are now proposed [SB06].

1.1.2 MPLS-based Techniques

MPLS relies on the label switching technique as in ATM or frame relay networks. In MPLS, the packets are assigned a label. The routers determine the outgoing interface to send the packets based on their label and eventually on the interface on which the packets are received. The forwarding decision does not rely on the

IP destination address. Before forwarding a packet, a router may change its label. This label is used in the forwarding process at the next router. A protocol, such as the Label Distribution Protocol (LDP) [ADF$^+$01] or the Resource reSerVation Protocol (RSVP) [BZB$^+$97], is used to exchange label information between neighboring routers.

MPLS is often used for traffic engineering purposes inside ISP networks [XHBN00], [ML05]. The main advantage of MPLS compared to classical IP routing is that MPLS can easily bypass the shortest path selected by IP routing. This is done by establishing Label Switched Paths (LSPs) between pairs of routers in the network.

A distinction is made between offline and online path computation for MPLS TE. The aim of offline path computation is to solve the traffic engineering problem statically. Based on a known traffic matrix, it is possible to compute an optimum layout of the LSPs in the network in order to spread the load among all the available links. This is more flexible than the tuning of the link costs in a pure IP network. With only the link costs, a change in the cost of one link may potentially affect the traffic distribution throughout the entire network. With MPLS, it is possible to force any flow of packets to follow a particular path. For example, it would be possible to force the packets from the $S_1 - D_1$ flow to follow the $R_{18} - R_{11} - R_{12} - R_{13} - R_{14} - R_{16} - R_{15}$ path while the packets from the $S_2 - D_1$ flow would follow the $R_{11} - R_{18} - R_{19} - R_{17} - R_{15}$ path in figure 1.1. This type of traffic distribution would not be possible by selecting the link costs in a pure IP network.

MPLS is also useful in a more dynamic environment where the traffic demand changes. In this case, MPLS is used in combination with intradomain routing protocols enhanced by TE extensions (e.g. OSPF-TE [KKY03] or ISIS-TE [SL04]). These protocols are extensions to the classical intradomain routing protocols that allow to distribute to all routers the entire topology of the network as well as the bandwidth and the reserved resources on each link. Based on this information, each router can determine the most heavily loaded links inside the network. In this case path computation is performed online. To understand online path computation and establishment of LSPs, let us consider figure 1.1 again. Let us assume that there are two important packet flows ($S_1 - D_1$ and $S_2 - D_2$). In addition, let us assume that links $R_{18} - R_{16}$ and $R_{16} - R_{15}$ are becoming congested while the other links are lightly utilized. Based on the information distributed by the TE extensions to the routing protocol, routers R_{11} and R_{18} are aware of the bandwidth reservation on the links. If these routers notice large flows from S_1 and S_2, they can redirect these flows over other links by establishing new LSPs or rerouting established LSPs. For this purpose, router R_{11} determines the set of constraints that must be fulfilled by the path and computes a path respecting the constraints for the LSP towards D_2. For example, R_{11} could compute a path that avoids the congested links ($R_{18} - R_{16}$ and $R_{16} - R_{15}$). If R_{11} knows that the $S_2 - D_2$ flow requires 100 Mbps on average, it could also compute a path where all the links have at least 100 Mbps of unreserved bandwidth. Then the signalling protocol, RSVP-TE (described in more details in chapter 3), establishes the LSP along the computed constrained path.

This signalling protocol can also reserve resources (e.g. bandwidth) along the path if required. An advantage of MPLS is that it is possible to reroute an existing LSP over another (e.g. less congested) path without breaking the existing LSP until the LSP is completely rerouted. Consequently, packet losses do not occur during the rerouting of an LSP contrary to the rerouting in a pure IP network [MD03]. This use of MPLS combined with TE extensions to routing protocols, constrained path computation and a signalling protocol allow the network to be traffic engineered in a more dynamic manner than with a pure IP solution. As a conclusion, the dynamic establishment of LSPs enables to distribute the traffic based on the current demand, to elaborate paths that are in accordance with the flows' requirements and also recover from failures in the network.

Another advantage of MPLS is its restoration capabilities [VPD04]. Restoration of MPLS LSPs is performed by setting up paths that avoid failed links or nodes. These paths may be setup after a failure occurred, as mentioned in the previous paragraph. In addition, they may be setup prior to failures. MPLS has the ability to protect an LSP against failures by means of pre-established LSPs. Consequently, it is possible to quickly recover in case of failure without recomputing a path for failed LSPs. When a failure occurs on the protected LSP, the traffic can be switched very quickly on the backup LSP without waiting for the convergence of the routing protocol as in pure IP solutions. Sharma et al. [SH03] give an overview of the different techniques to protect an LSP. All these techniques rely on the ability to select a path according to some constraints. A first technique aims at protecting an entire LSP with a single backup LSP. In this case, the backup LSP is disjoint from the protected LSP. This type of protection is called global repair. When a failure occurs, a failure indication message is sent upstream along the protected LSP to force the traffic to be switched on the backup LSP at the head-end node of both LSPs. In this case, the restoration time is comparable to the propagation delay inside the network. A second technique is to protect individual resources (i.e. link or routers). We talk about local repair. In this case, multiple LSPs may be needed to protect the resources of a single LSP. These LSPs may be dedicated to the protection of a single LSP. They are called detour LSPs [PSA05]. Or, they may be used for the protection of multiple LSPs. These backup LSPs are then called bypass tunnels. For example, to protect the $R_{18} - R_{19} - R_{17} - R_{15}$ path from link failure, three backup LSPs would be established, one on the $R_{18} - R_{16} - R_{19}$ path, one on the $R_{19} - R_{16} - R_{15}$ path and the other on the $R_{17} - R_{19} - R_{16} - R_{15}$ path. If a failure is detected by router R_{18} on the link $R_{18} - R_{19}$, then R_{18} redirects immediately the packets of the $R_{18} - R_{19} - R_{17} - R_{15}$ path through the backup LSP established via R_{16}. These packets would reach R_{19} as if they used the $R_{18} - R_{19}$ link and would be sent downstream as usual. In this case, the restoration time depends mainly on the time required to detect the failure. It is faster than with global protection. Once the traffic has been rerouted over a backup LSP, a failure notification message can be sent to the head-end of each rerouted LSP to allow it to recompute a new optimized path taking the failure into account.

In this section, we have underlined the advantages of MPLS TE compared to the tuning of the IGP weights. These advantages are the facility in avoiding congested resources, the provision of QoS and the capability to protect traffic against failures. MPLS is a powerful tool to engineer the traffic inside an AS. The engineering of the traffic inside an AS and the provision of QoS inside an AS, with MPLS, are widely addressed in the literature [FE05]. Now, SPs require the same functionalities for the traffic that crosses AS boundaries [RV05]. The objective in this thesis it to provide the necessary building blocks for MPLS inter-AS TE including QoS.

1.2 List of Contributions

In this section we provide a list of the main contributions of the thesis. These contributions consist of:

- Active measurements of BGP TE techniques
- Extensions to RSVP-TE for inter-AS LSP establishment and protection
- Study of the impact of BGP routes on the computation of inter-AS constrained paths
- Modifications, implementation and evaluation of distributed path computation techniques

1.3 Content of the Thesis

Interest on Traffic Engineering has been mostly focused on intra-domain issues such as the distribution of traffic inside a domain and the provision of QoS to users connected to a single domain [XHBN00, AKK$^+$00, EJLW01, GCL04]. Later, the research community started working on the distribution of traffic on links peering with other domains [Bar02, UBQ03, ACK03, dBL03, FBR03, QB05] . For this purpose, many Internet Service Providers tune the configuration of the Border Gateway Protocol (BGP) on their routers on a trial and error basis [ACE$^+$02]. Content providers often need to control their outgoing traffic while access providers need to control their incoming traffic. In the first chapter, we show, by means of measurements, that controlling the flow of the incoming traffic is a difficult problem. For this purpose, we rely on detailed active measurements to show the limitations of AS-Path prepending and the BGP communities. We show that the difficulty of controlling the flow of the incoming traffic lies in the difficulty of predicting which BGP route will be selected by distant ASs.

MultiProtocol Label Switching (MPLS) is used inside large ISP networks to provide services with stringent Service Level Agreements such as Virtual Private Networks (VPNs). Customers are now urging ISPs to provide such services across interdomain boundaries [RV05]. This requires the ability to establish interdomain

MPLS Label Switched Paths (LSPs) with constraints. In the second chapter of this thesis, we present extensions to the RSVP-TE signaling protocol enabling the establishment of interdomain LSPs. This chapter contains the state of the art in RSVP-TE interdomain LSP establishment for packet-switched networks as well as personal contributions. We propose extensions enabling the establishment of LSPs toward prefix and AS destinations for TE purposes. We provide a way to hide confidential topology information while still being able to locally protect links, nodes and Shared Risk Link Groups along the LSPs against single failures.

Up to now, the literature has mostly focused on mechanisms to compute constrained LSPs inside a single AS. When traffic engineered LSPs with QoS must be terminated at a router in another Service Provider (SP) network the selection of the path becomes a problem. There is no node that knows the entire topology. This is because the networks exchange routing information by using the Border Gateway Protocol (BGP). In contrast to OSPF-TE/ISIS-TE where each node has a map of the SP network, BGP does not distribute complete topology, delay and bandwidth information. In this thesis, we propose distributed techniques under development at the IETF for the computation of interdomain constrained LSPs. We finalize these solutions and provide an implementation of these techniques in a simulator.

Then, we investigate the complexity of establishing end-to-end interdomain LSPs with QoS constraints, based on the BGP routes locally available at a router. We explain the main issues of relying on BGP for the computation of interdomain constrained paths. To illustrate our point, we compare two LSP establishment techniques. Our benchmark technique is centralized and assumes the complete knowledge of the intradomain topologies. The second path computation technique is distributed and relies on the BGP routes locally available at each router. Our simulations confirm that in designing BGP-based interdomain LSP establishment techniques the amount and the quality of the BGP routes that serve in the computation play an important role. We recommend to introduce Path Computation Elements (PCEs) inside SP networks, to collect all the BGP routing information available in a SP network and to perform the distributed computation at these nodes.

Due to the lack of QoS information concerning the BGP routes, we rely on heuristics to estimate the QoS of the paths. We develop distributed path computation techniques that do not require changes to the BGP routing protocol. We evaluate the performance of these techniques in the computation of maximum delay and minimum bandwidth guaranteed LSPs in addition to maximum delay constrained LSPs. Our heuristics perform well in both cases. Moreover, we show that the amount of signaling required by the distributed path computation techniques does not have to be high in order to achieve good results.

Because SPs are free to use their own intradomain Traffic Engineering (TE) algorithms without coordination with their peers, we evaluate by means of simulations the interactions of an intradomain TE algorithm, DAMOTE [BML03b], and the computation of interdomain LSPs with QoS requirements. We show that such a TE algorithm that aims at distributing the bandwidth inside an AS performs well in the provision of LSPs that are only subject to a bandwidth reservation. In this case, we have observed and explained why short end-to-end delay paths are found. In addition, we have seen that if such a TE algorithm that aims at distributing the traffic inside an AS also tries to minimize the length of the computed paths, it is appropriate for the provisioning of interdomain LSPs with delay and bandwidth constraints.

1.4 Publication List

The major part of the contributions of this thesis have already been published. Here is a list of the different journal and conference papers related to my thesis:

- Cristel Pelsser and Olivier Bonaventure. *Path Selection Techniques to Establish Constrained Interdomain MPLS LSPs*. in Proc. of IFIP International Networking Conference,LNCS 3976, May 2006.
- Cristel Pelsser. *Using virtual coordinates in the establishment of inter-domain LSPs*. In Proc. of CoNEXT 2005 Student Workshop, October 2005.
- Bruno Quoitin, Cristel Pelsser, Olivier Bonaventure and Steve Uhlig. *A performance evaluation of BGP-based traffic engineering*. International Journal of Network Management, Volume 15 , Issue 3, May 2005.
- Steve Uhlig, Cristel Pelsser, Bruno Quoitin and Olivier Bonaventure. *Vers des réflecteurs de routes plus intelligents*. Colloque Francophone sur l'Ingénierie des Protocoles (CFIP 2005), March-April 2005. (in French) **Best Paper Award**
- Cristel Pelsser, Steve Uhlig and Olivier Bonaventure. *On the difficulty of establishing interdomain LSPs*. in Proc. of 2004 IEEE International Workshop on IP Operations and Management (IPOM 2004), October 2004.
- Cristel Pelsser and Olivier Bonaventure. *Extending RSVP-TE to support Inter-AS LSPs*. in Proc. of 2003 Workshop on High Performance Switching and Routing (HPSR 2003), June 2003.
- Bruno Quoitin, Steve Uhlig, Cristel Pelsser, Louis Swinnen and Olivier Bonaventure. *Interdomain Traffic Engineering with BGP*. IEEE Communications Magazine Internet Technology Series,Volume 41, Issue 5, May 2003.

Several contributions were also submitted to the IETF. Here is a list of IETF Internet Drafts concerning extensions to RSVP-TE and the limitations of BGP:

- Cristel Pelsser, Steve Uhlig and Olivier Bonaventure. *Limitations induced by BGP on the computation of interdomain LSPs*. work in progress, draft-pelsser-bgp-pce-00.txt.

- Stefaan De Cnodder and Cristel Pelsser. *Protection for inter-AS MPLS tunnels*. work in progress, draft-decnodder-ccamp-interas-protection-00.txt, July 2004.

- Cristel Pelsser and Olivier Bonaventure. *RSVP-TE extensions for interdomain LSPs*. work in progress, draft-pelsser-rsvp-te-interdomain-lsp-00.txt, October 2002.

1.5 Acknowledgements

Several persons directly or indirectly contributed to the work presented in this thesis. The intention of this section is to thank all these persons. I apologize in advance to the persons that are not listed here. So many of you contributed to what I am now, how I apprehend the world and my research. Thank to all of you.

First of all, I want to thank my advisor, Olivier Bonaventure. Olivier is a very good and motivated advisor. He is attentive to the work of his researchers, always trying to push them beyond their limits and giving numerous advises. I learned a lot from his pragmatic view of networking. I thank him for giving me the opportunity to work with him.

Stefaan De Cnodder was my advisor during my master thesis. Together with Olivier, they introduced me to research. The work on protection of LSPs, in section 3.7, is a joint work with Stefaan. Stefaan has been a critical reviewer of chapters 2, 3 and 5. I thank Stefaan for our enriching collaboration.

I also thank my colleague researchers in the team of Olivier. I acknowledge Steve Uhlig for the endless discussions we had in the last years. Steve gave useful comments on all my papers and he participated in the writing of the paper on chapter's 5 content. His questions stimulated my reflection and broadened my point of view.

Bruno Quoitin is the author of the C-BGP simulator that I have used extensively during my thesis. I an indebted to him for developing and supporting this tool. Bruno also provided me with various scripts, a few latex and administration tips.

Sébastien Tandel contributed to the work on BGP measurements presented in chapter 2. Cédric de Launois provided the algorithms to compute the vivaldi coordinates (see section 6.1.2). Pierre François's insatiable curiosity lead to many discussions on the computation of interdomain constrained paths and its implications with regard to policies. Louis Swinnen, Pierre Reinbold and Virginie Van den Schrieck were or are still colleagues. Each of my colleague had at least once the occasion to review and comment my papers. My colleagues contributed to a good working environment. Over the years, they became friends.

I thank Simon Balon and Gael Monfort for the collaboration concerning interactions with the DAMOTE algorithm distributed with the TOTEM toolbox. They were enthusiastic and very prompt to make changes and comments on the direction to take. I thank Guy Leduc, their advisor, for the comments he made during meetings all along my thesis.

I am grateful to Jean-Louis Leroux, Jean-Philippe Vasseur and Sharad Agarwal for their comments on different aspects of my research.

I wish also to thank Jan Torrele from Belnet and the operators from the anonymous Belgian commercial ISP. Our measurements could not have been performed without them.

The work in this thesis would not exist without the support, love, education provided by my parents and the continuous encouragements of my companion. Without the patience of my host families in the USA to teach me English, I would certainly have a much harder time to write this thesis and you to read it.

This work was supported by the European Commission within the IST ATRIUM project, by the Walloon Government (DGTRE) under the WIST TOTEM project and partially by the European Commission under the E-Next project FP6-506869.

Chapter 2

Interdomain TE Techniques

In the previous chapter, we gave an overview of the techniques used to engineer the traffic inside an AS. Now, we interest ourselves to the techniques available today to engineer the traffic crossing multiple ASs. These techniques rely on the Border Gateway Protocol.

Interdomain traffic engineering covers the various techniques that enable ASs to control their interdomain traffic. Despite its importance, as explained in [ACE$^+$02], interdomain traffic engineering is today more an art than a science. Furthermore, for many Service Providers (SPs), interdomain traffic engineering is still done on a trial and error basis. There have been very few detailed studies of the performance of those techniques. Several techniques have been proposed to allow a content provider to optimize its outgoing interdomain traffic (see [Bar02, UBQ03] and the references therein). However, detailed studies on the ability of using BGP to engineer the traffic entering a domain were performed later. Among these studies there is the work of Gao [GDZ05], Wang [WCCL05] and Lo [LC05].

The Internet is composed of more than 23000 distinct ASs operated by Internet Service Providers (ISPs), corporations or universities. These networks exchange reachability information by means of the Border Gateway Protocol (BGP) [RLH06, Hal00]. BGP is thus an important building block of the Internet. The current interdomain TE techniques rely on this protocol, as we will see in this chapter. However, BGP is a complex protocol which enforces various economical relationships among the ASs.

There are two types of ASs in today's Internet. A stub AS is an AS that sends or receives IP packets, but does not transit packets. A transit AS is an AS that agrees to transit IP packets from one of its neighbors to another neighbor. [Gao00] has identified two main types of relationships enforcing this classification. A *customer-to-provider* relationship is used when a customer AS buys connectivity from a provider AS. A *peer-to-peer* relationship is used when the connection cost is shared by the two ASs.

A stub AS is connected to one or several provider ASs that the stub uses as transit to send IP packets to any destination. The tendency for a stub AS is to be multiply-connected [SARK02] to different providers for redundancy and Traffic Engineering (TE) purposes. Today, around 60% of customer ASs are multi-homed and this percentage is increasing [ACK03].

For stub ASs, which represent the majority of the ASs today (70% according to [Hus06a]), controlling how the Internet traffic enters or leaves their network is an important problem. For instance, stub domains which provide content are interested in controlling the flow of their outgoing traffic. Indeed, they want to optimize the way information reaches their customers.

In this chapter, we focus on the control of the incoming traffic by multi-homed stub domains, a common operational problem. We rely on measurements to explain why this control of the flow of the incoming traffic is difficult. Our measurements were performed in 2003. To our knowledge, it is the first time such measurements have been performed in the global Internet.

The remainder of this chapter is organized as follows. First, we summarize the main BGP traffic engineering techniques in section 2.1. Then, we present, in section 2.2, our measurement-based evaluation of the most common TE technique, AS-Path prepending. This is, to our knowledge, the first analysis of such active measurements. Second, we present measurements with a second technique that relies on BGP communities. Finally, we provide an overview of the TE techniques that do not directly rely on BGP, in section 2.3.

2.1 BGP TE Techniques

Inter-domain traffic engineering is concerned with the performance optimization for traffic that originates in one administrative domain and terminates in a different one [ACE+02].

Most of the current work on interdomain Traffic Engineering (TE) is based on BGP. It makes use of the BGP attributes in route advertisements to influence the selection and readvertisement of the BGP routes. Among the contributions on interdomain TE based on BGP we can cite [UQ05, TGRR05, UBQ03, FBR03] for their work on outbound TE and [GDZ05, LC05, WCCL05] for their work on inbound TE.

The BGP traffic engineering techniques aim at modifying the BGP routes selected by the routers in order to change the paths followed by the traffic. A router that receives several BGP routes for a prefix, selects one of these routes by means of the decision process illustrated in table 2.1. The rules of the decision process are applied in sequence to the set of available routes until a single route is obtained. At each step only the routes with the preferred value for the current criteria are selected for the next step. For example, in the first step of the decision process, all the routes that do not have the highest value for the `LOCAL_PREF` attribute are removed from consideration. If there is only one route with the highest `LOCAL_PREF` than this route is the best route. Subsequently, it is inserted in the routing table. However, if there are several routes with the highest `LOCAL_PREF` value, all these routes are considered in the next step of the decision process.

Step	Rule
1	Prefer routes with highest `LOCAL_PREF`
2	Prefer routes with shortest `AS-PATH`
3	Prefer routes with lowest `MED`
4	Prefer eBGP over iBGP routes
5	Prefer routes with lowest IGP cost to the next-hop
6	Tie breaking rules

Table 2.1: Simplified BGP decision process

BGP TE can be performed by avoiding to redistribute certain routes as well as by changing the values of certain attributes to favor or penalize certain routes. This is accomplished by means of filters. When a route is received, a router applies import filters to discard some routes or change the attributes of certain routes. Then, the decision process is applied on the set of routes that passed the import filters. Finally, export filters are applied to the routes selected by the decision process before advertising them to neighboring BGP peers. These filters may also discard or modify the routes.

The filters are also used to implement the peering relationships of ASs. [Gao00] distinguishes two major types of peering: the *customer-provider* and the *peer-peer* peering relationships. The first type concerns a customer AS connected to a provider AS. The customer buys connectivity from the provider. The provider supplies a transit service to its customer. It carries its traffic to any destination and all the traffic destined to the customer. On the other hand, two ASs in a peer-peer relationship share the cost of their connection. They mutually agree to carry the traffic exchanged between them or their respective clients. They provide a limited transit service to their peer. The BGP policies are the set of rules that define the routes that can be redistributed to a given peer and the preferences of the routes based on type of peering on which the routes are learned. Customer-provider and peer-peer relationships imply the definition of BGP policies. These policies are implemented through the use of the `LOCAL_PREF` and the `communities` attributes.

From these types of peerings a hierarchy of ASs is observed. At the top of the hierarchy there are the Tier-1 Internet Service Providers (ISPs) that are providers or peers of other ASs but are not customers. At the bottom of the hierarchy we have the stub ASs. These ASs are customers or peers of other ASs but they do not provide transit service on behalf of other ASs. They are not providers for other ASs. We will see that the impact of BGP TE techniques is different depending on the rank of the ASs in the hierarchy.

In the remaining of this section, we present the different BGP TE techniques. We distinguish the techniques that enable to engineer the traffic that leaves an AS, in section 2.1.1, from the techniques that influence the way traffic enters an AS, in section 2.1.2.

2.1.1 Engineering the Outgoing traffic

To engineer its outgoing traffic an AS needs to be able to influence the routes that are selected by each router inside its own AS. This can be done by means of the attributes implied in the first rules of the decision process.

The `LOCAL_PREF` attribute may be used to engineer the outgoing traffic. Because its value is used in the first steps of the decision process, a high value favors the consideration of the route in the latter steps of the decision process while a low value leads to the removal of the route from consideration for the best route selection process.

The local preference, the `LOCAL_PREF` attribute, of a route may be set by the import filters. This value is redistributed with the route inside the AS. Thus, a route that is assigned a high `LOCAL_PREF` value by a one router in the AS will have a high preference at all routers inside the AS, if this value is not overwritten by the import filters of the other routers. However, in this case, the configuration of the different routers in the AS is not consistent and routing loops could occur.

Figure 2.1 illustrates the use of `LOCAL_PREF` to engineer the outgoing traffic. The routes selected as best routes are preceded by "$>$". Suppose that a large amount of traffic is originated in $AS1$ toward prefix $P1$ and $P2$, with an approximately equal load toward each prefix. In the original situation, the traffic toward both prefixes transits through $AS5$ because the `AS-path` is shorter. This is shown at the top of the figure, by the BGP Adjacency Routing Information Base that contains the routes (Adj_RIB_In) received by $R11$ on each BGP session after the application of the import filters. If the administrator of $AS1$ wants to balance the outgoing traffic on both interdomain links, a `LOCAL_PREF` of 200 can be assigned by $R12$ to the route announcing prefix $P2$. The Adj_RIB_In at the bottom of figure 2.1 shows that the traffic for prefix $P2$ now crosses $AS2$ and $AS3$ while the traffic toward $P1$ remains on link $R13 - R51$.

2.1.2 Engineering the Incoming traffic

There are five techniques to engineer the traffic entering an AS. The first technique is called `AS-path` prepending. It consists in virtually decreasing the quality of a route by increasing the length of the `AS-path`. A router that performs `AS-path` prepending adds the AS Number (ASN) of its AS inside the `AS-path`. The length of the `AS-path` is used in the decision process because it was assumed to give an indication of the quality of the route. However, [McM99] and [HFP$^+$02] have shown that this assumption is not always true.

In figure 2.2, $AS1$ wants to engineer its incoming traffic. It receives a lot of traffic from $AS4$ and $AS6$ toward prefix $P1$. However, in the basic situation both ASs use link $R51 - R13$ to join $P1$. This is shown in the Adj_RIB_In of $R41$ and $R61$ in the left of figure 2.2. If $R13$ in $AS1$ prepends its ASN two times to the

R11

Normal situation

	NH	Prefix	AS-path	LocPref
	R21	P1	AS2-AS3-AS4	100
	R21	P2	AS2-AS3-AS6	100
>	R51	P1	AS5-AS4	100
>	R51	P2	AS5-AS6	100

Local preference

	NH	Prefix	AS-path	LocPref
	R21	P1	AS2-AS3-AS4	100
>	R21	P2	AS2-AS3-AS6	200
>	R51	P1	AS5-AS4	100
	R51	P2	AS5-AS6	100

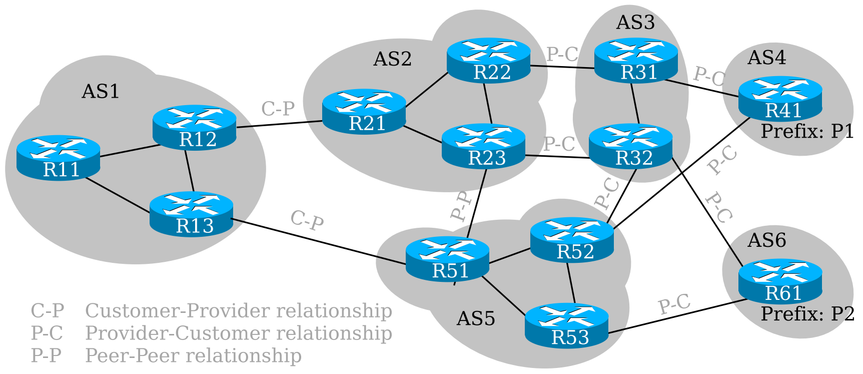

AS1
AS2
AS3
AS4
AS5
AS6
R11
R12
R13
R21
R22
R23
R31
R32
R41
R51
R52
R53
R61
Prefix: P1
Prefix: P2
C-P
P-C
P-P
C-P Customer-Provider relationship
P-C Provider-Customer relationship
P-P Peer-Peer relationship

Figure 2.1: Outgoing Traffic Engineering

`AS-path` before advertising prefix $P1$ to $R51$, both $AS4$ and $AS6$ will use link $R21 - R12$ to reach $P1$.

R41

Normal situation

	NH	Prefix	AS-path
	R31	P1	AS3-AS2-AS1
>	R52	P1	AS5-AS1

Prepend AS-path twice

	NH	Prefix	AS-path
>	R31	P1	AS3-AS2-AS1
	R52	P1	AS5-AS1-AS1-AS1

R61

Normal situation

	NH	Prefix	AS-path
	R32	P1	AS3-AS2-AS1
>	R53	P1	AS5-AS1

Prepend AS-path twice

	NH	Prefix	AS-path
>	R32	P1	AS3-AS2-AS1
	R53	P1	AS5-AS1-AS1-AS1

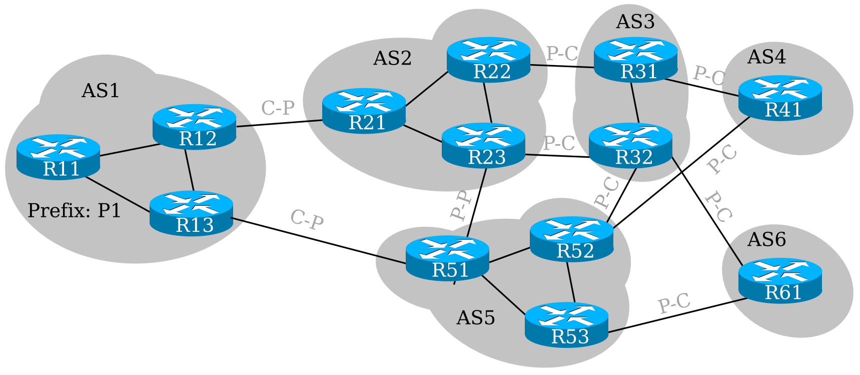


Figure 2.2: `AS-path` prepending: example 1

In figure 2.3, we see that if $AS1$ advertises more than one prefix, the load can better be distributed on the two incoming links. The `AS-path` in the route for one prefix is prepended two times when announced to $R51$ while the route for the other prefix is not prepended.

R41

	Normal situation				Prepend AS-path twice		
	NH	Prefix	AS-path		NH	Prefix	AS-path
	R31	P1	AS3-AS2-AS1		R31	P1	AS3-AS2-AS1
>	R52	P1	AS5-AS1	>	R52	P1	AS5-AS1
	R31	P2	AS3-AS2-AS1	>	R31	P2	AS3-AS2-AS1
>	R52	P2	AS5-AS1		R52	P2	AS5-AS1-AS1-AS1

R61

	Normal situation				Prepend AS-path twice		
	NH	Prefix	AS-path		NH	Prefix	AS-path
	R32	P1	AS3-AS2-AS1	>	R32	P1	AS3-AS2-AS1
>	R53	P1	AS5-AS1		R53	P1	AS5-AS1
	R32	P2	AS3-AS2-AS1	>	R32	P2	AS3-AS2-AS1
>	R53	P2	AS5-AS1		R53	P2	AS5-AS1-AS1-AS1

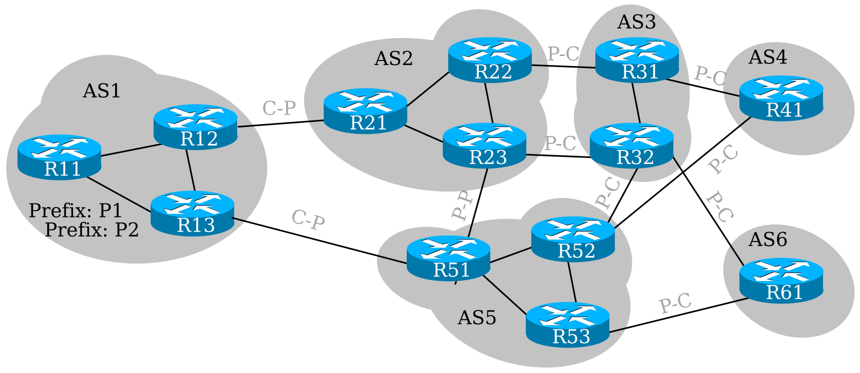


Figure 2.3: `AS-path` prepending: example 2

The second technique for an AS to engineer its incoming traffic makes use of the `MED` attribute. The `MED` is used to engineer the traffic between two domains that are multiply connected with several BGP peering sessions[1]. It is not used in the case of stub ASs multi-homed to different providers, which is a very common situation today [ACK03]. Figure 2.4 illustrates the use of the `MED` attribute. The objective of $AS1$ in using the `MED` attribute is to reduce the cost to transport the traffic inside its own network. However, as we will see later, this often leads to an increase in this cost for the peering AS ($AS2$, in figure 2.4). Thus, to use this technique both ASs need to agree on the use of `MED` attribute. Otherwise, $AS2$ could overwrite this attribute by means of the input filters and $AS1$'s attempt to engineer its incoming traffic would fail.

In figure 2.4, router $R11$ announces prefix $P1$ while $R13$ announces $P2$. The links are labeled with their IGP cost. Both $R12$ and $R14$ readvertise prefixes $P1$ and $P2$ outside the AS. In the original situation, $R21$ and $R23$ in $AS2$ receive two routes for each prefix. One of these routes is learned from their respective external peer in $AS1$. The other route is learned from the other peer with $AS1$. Both $R21$ and $R23$ select the route learned from the external peer as the best route because both routes have the same local preference, due to the same BGP policies on the

[1]The `MED` attributes of routes learned from different ASs are not compared, in general.

Normal situation

R22

	NH	Prefix	AS-path	MED
>	R21	P1	AS1	0
	R23	P1	AS1	0
>	R21	P2	AS1	0
	R23	P2	AS1	0

R24

	NH	Prefix	AS-path	MED
>	R21	P1	AS1	0
	R23	P1	AS1	0
>	R21	P2	AS1	0
	R23	P2	AS1	0

MED

R22

	NH	Prefix	AS-path	MED
>	R21	P1	AS1	1
	R23	P1	AS1	11
	R21	P2	AS1	11
>	R23	P2	AS1	1

R24

	NH	Prefix	AS-path	MED
>	R21	P1	AS1	1
	R23	P1	AS1	11
	R21	P2	AS1	11
>	R23	P2	AS1	1

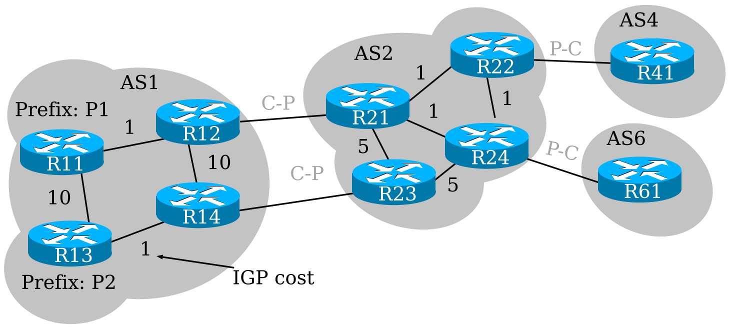


Figure 2.4: Multi-Exit Discriminator

two peering sessions, the same `AS-path` length and there is no `MED`. These routes are also announced to $R22$ and $R24$. $R22$ and $R24$ learn the routes for $P1$ and $P2$ via iBGP. For each prefix, they select the route with the lowest IGP cost to the NH because the previous rules in the decision process do not allow to eliminate routes from the best route selection process. Because router $R21$ is the nearest NH for both $R22$ and $R24$, they select the route learned from $R21$ for both prefixes. As a consequence, the traffic toward $P1$ and $P2$ enters $AS1$ via link $R21 - R12$.

If $R12$ and $R14$ set the `MED` value of the routes to the IGP cost of the path toward the NH, the routers inside $AS2$ now select their best route based on the `MED` value. For prefix $P1$, they prefer the route learned at $R21$ from $R12$. Yet, they select the route announced to $R23$ by $R14$ for prefix $P2$. Thus, if both prefixes attract the same amount of traffic, the load will be balanced on links $R21 - R12$ and $R23 - R14$.

A third technique that is often used to control the flow of the incoming traffic is to rely on BGP communities [BQ03]. BGP communities are special values that are attached to BGP advertisements and used to request remote routers to perform some actions. The following traffic engineering actions are often supported:

- do not announce the route : in this case the route with the associated community should not be announced to the specified peers
- prepend n times when announcing the route : the AS-path of the route with the associated community will be prepended n times when it is announced to the specified peers

- specify the value of the local preference to be used by the router that receives the route [CB96].

Figure 2.5 illustrates a simple use of the BGP communities attribute. In this example, the administrator of $AS5$ defines a value for the BGP communities attribute that can be used by its clients to avoid the announcement of a route to a group of peers. Here this group corresponds to $AS4$ and $AS6$. If $AS1$ attaches the community to the announcement of $P2$ toward $R51$, $R52$ and $R53$ will not announce the route to $AS4$ and $AS6$. These ASs now only have one route toward $P2$. As a consequence, the traffic to $P2$ is forced on link $R21 - R12$. However, the traffic to $P1$ still flows through $AS5$ and is carried on link $R51 - R13$. The drawback of communities of type “do not announce” is that certain routes are not distributed. Thus, if a failure occurs, the prefix may not be reachable anymore.

Normal situation

R41

	NH	Prefix	AS-path
	R31	P1	AS3-AS2-AS1
>	R52	P1	AS5-AS1
	R31	P2	AS3-AS2-AS1
>	R52	P2	AS5-AS1

R61

	NH	Prefix	AS-path
	R32	P1	AS3-AS2-AS1
>	R53	P1	AS5-AS1
	R32	P2	AS3-AS2-AS1
>	R53	P2	AS5-AS1

Do not announce

R41

	NH	Prefix	AS-path
	R31	P1	AS3-AS2-AS1
>	R52	P1	AS5-AS1
>	R31	P2	AS3-AS2-AS1

R61

	NH	Prefix	AS-path
>	R32	P1	AS3-AS2-AS1
	R53	P1	AS5-AS1
>	R32	P2	AS3-AS2-AS1

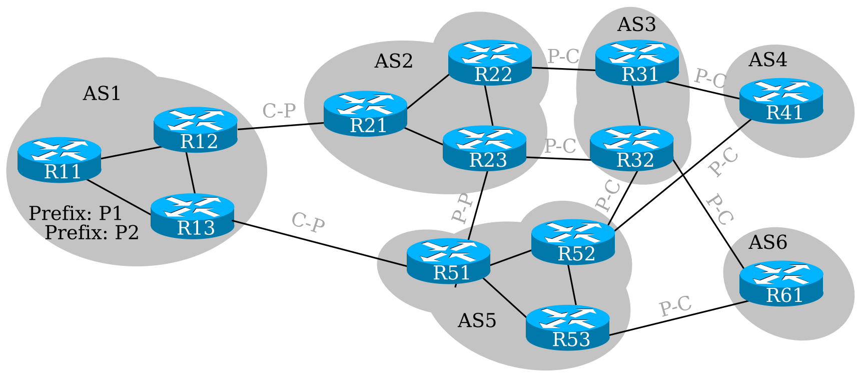


Figure 2.5: Do not announce community

In figure 2.6, we show the routes selected by $R41$ and $R61$ if a “prepend twice” community replaces the “do not announce” community of the previous example. The target of this community is still $AS4$ and $AS6$. We note that $R41$ and $R61$ have two routes for $P2$ instead of a single route, with the “do not announce” community. $AS5$ has prepended its ASN twice to the `AS-path` before advertising the route for $P2$ to $R41$ and $R61$. Thus, the route is less preferred but it is available for use in case of a failure impacting the best route.

The technique called “selective advertisements” consists in announcing different (non overlapping) prefixes to different peers. For example, in figure 2.7, $AS1$

R41

Normal situation				prepend twice			
	NH	Prefix	AS-path		NH	Prefix	AS-path
	R31	P1	AS3-AS2-AS1		R31	P1	AS3-AS2-AS1
>	R52	P1	AS5-AS1	>	R52	P1	AS5-AS1
	R31	P2	AS3-AS2-AS1	>	R31	P2	AS3-AS2-AS1
>	R52	P2	AS5-AS1		R52	P2	AS5-AS5-AS5-AS1

R61

Normal situation				prepend twice			
	NH	Prefix	AS-path		NH	Prefix	AS-path
	R32	P1	AS3-AS2-AS1	>	R32	P1	AS3-AS2-AS1
>	R53	P1	AS5-AS1		R53	P1	AS5-AS1
	R32	P2	AS3-AS2-AS1	>	R32	P2	AS3-AS2-AS1
>	R53	P2	AS5-AS1		R53	P2	AS5-AS5-AS5-AS1

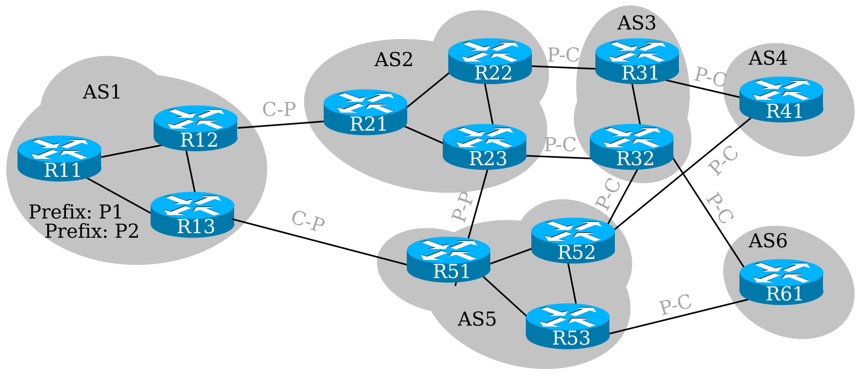


Figure 2.6: Prepend twice community

only announces prefix $P1$ to $AS5$ and $P2$ to $AS2$. As a consequence $P1$ can only be reached through link $R51 - R13$ and $P2$ through link $R21 - R12$. We note that if one of these two links fails either $P1$ or $P2$ are not reachable without a BGP configuration change.

The last BGP traffic engineering technique that we present here concerns the advertisement of a prefix to all the external peers and the advertisement of more specific prefixes[2] to a subset of the external peers. This technique is illustrated in figure 2.8. In this figure, $AS1$ first announces one prefix, $P1(/23)$. As a result, all the incoming traffic is received on link $R51-R13$. However, when $AS1$ announces $P1(/23)$ on the two external peering sessions and the more specific prefix $P2(/24)$ only to $AS2$, the traffic destined to $P2(/24)$ enters $AS1$ via link $R21 - R12$ while the traffic destined to $P1(/23)$ but not $P2(/24)$ enters the AS via the other interdomain link. The drawback of this technique is that the number of routes to maintain in the BGP routing tables increases, as it is shown by [Hus01]. In order to avoid an increase in the size of their BGP routing tables the operators usually filter out prefixes smaller than the smallest prefixes allocated by regional Internet registries [BBGR01]. For example, the smallest IPv4 prefixes allocated by RIPE have a 21 bits mask [RIP06]. In addition, we note that a failure of link $R21 - R21$ does not have an impact on the reachability of prefix $P2$. Upon the failure of this

[2]A more specifi c prefi x represents a set of addresses that are all contained in the set designated by another prefi x, a less specifi c prefi x.

Normal situation

		NH	Prefix	AS-path
R41		R31	P1	AS3-AS2-AS1
	>	R52	P1	AS5-AS1
		R31	P2	AS3-AS2-AS1
	>	R52	P2	AS5-AS1
R61		R32	P1	AS3-AS2-AS1
	>	R53	P1	AS5-AS1
		R32	P2	AS3-AS2-AS1
	>	R53	P2	AS5-AS1

Selective advertisements

		NH	Prefix	AS-path
R41		R31	P1	AS3-AS5-AS1
	>	R52	P1	AS5-AS1
	>	R31	P2	AS3-AS2-AS1
		R52	P2	AS5-AS2-AS1
R61	>	R32	P1	AS3-AS5-AS1
		R53	P1	AS5-AS1
	>	R32	P2	AS3-AS2-AS1
		R53	P2	AS5-AS2-AS1

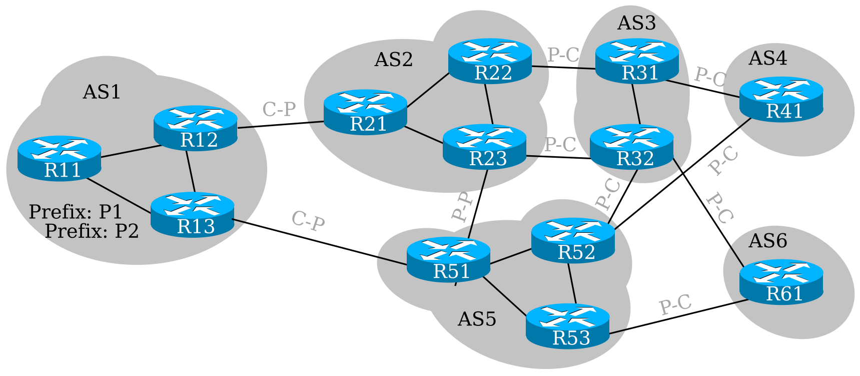


Figure 2.7: Selective advertisements

link, the route for $P2$ is withdrawn but the route for $P1$ is then used to route the traffic destined to $P2$.

2.2 Evaluation of BGP TE Techniques

After a presentation of several BGP techniques to engineer the traffic entering an AS, in section 2.1.2, we evaluate two of these techniques in this section: `AS-path` prepending and BGP communities. For, our evaluation we take the case of a dual-homed stub[3] AS and perform active measurements. Our stub is connected to two different providers. According to [SARK02] stub ASs represent 82% of the ASs. Moreover, at least 60% of the stub ASs are multi-homed to two or more providers, as stated by [ACK03]. In our measures, we do not consider the technique based on the `MED` because it applies to ASs multiply connected to the same provider. Furthermore, the `MED` may cause persistent routing oscillations [GW02a]. Moreover, we do not measure the impact of the technique relying on more specific prefixes since as [BBGR01] we do not consider this as a suitable technique.

In this section, we first present the methodology used for the measurements. Then, we analyze the results of the measurements performed with `AS-path`

[3]A stub AS is an AS that sends or receives traffi c but does not transit traffi c on behalf of other ASs.

R41

Normal situation

	NH	Prefix	AS-path
	R31	P1	AS3-AS2-AS1
>	R52	P1	AS5-AS1

More specific prefixes

	NH	Prefix	AS-path
	R31	P1	AS3-AS2-AS1
>	R52	P1	AS5-AS1
>	R31	P2	AS3-AS2-AS1
	R52	P2	AS5-AS2-AS1

R61

Normal situation

	NH	Prefix	AS-path
	R32	P1	AS3-AS2-AS1
>	R53	P1	AS5-AS1

More specific prefixes

	NH	Prefix	AS-path
>	R32	P1	AS3-AS2-AS1
	R53	P1	AS5-AS1
>	R32	P2	AS3-AS2-AS1
	R53	P2	AS5-AS2-AS1

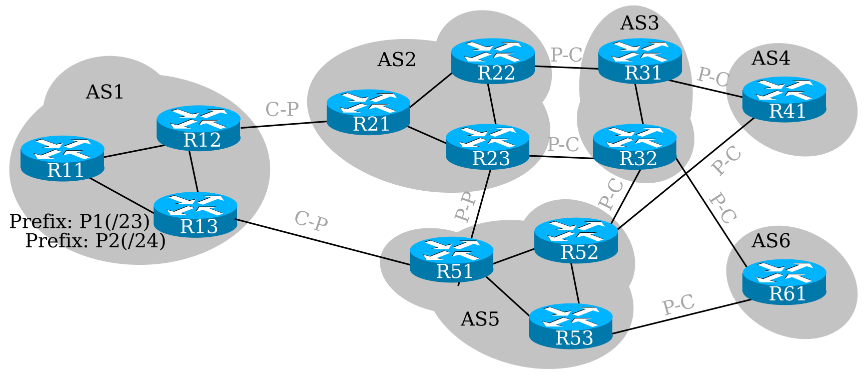


Figure 2.8: More specific prefixes

prepending, in section 2.2.2, and with the communities, in section 2.2.3. The measurements took place between March 12th, 2003 and May 14th, 2003. To our knowledge it is the first time such Internet-wide active measurements were performed.

2.2.1 Methodology

Our measurement study of AS-Path prepending was carried out by using a temporary stub AS number, AS2111, that was connected to two distinct providers: Belnet (AS2611), the Belgian Research Network, and a Belgian commercial ISP[4] (Be_ISP). At that time, Belnet was connected to 147 direct peers at several interconnection points and had two providers : TeliaNet (AS1299) and Level3 (AS3356). Belnet was also attached to the European research network, GEANT (AS20965). The Be_ISP was connected to 22 peers and had one provider. Figure 2.9 describes the AS-level topology as inferred from the BGP tables received from our two providers. This figure also shows the direct links between those providers. However, it should be noted that all these providers are also connected to larger ISPs.

We configured the BGP router of AS2111 to advertise a single `/20` prefix to its two providers. Note that by using a `/20` prefix, we ensure that our prefix was

[4]We unfortunately cannot reveal the identity of this ISP.

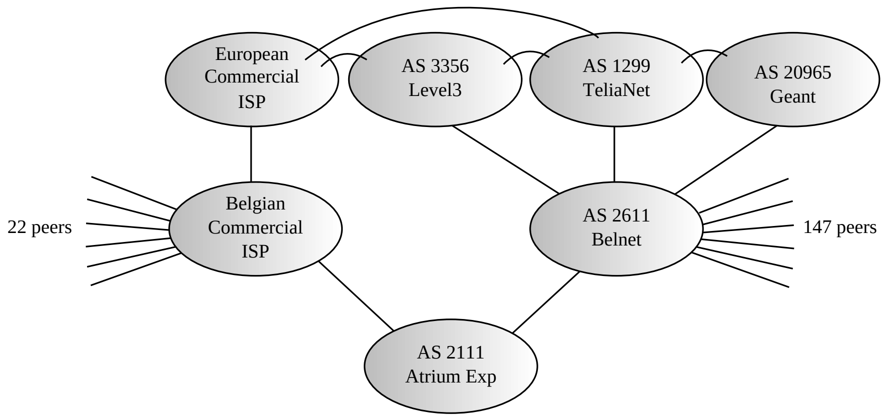


Figure 2.9: View from AS2111

not dropped by prefix length filters implemented by distant ISPs.

Since our aim is to control the flow of the incoming traffic, the first step in our study consisted in generating traffic from as many sources as possible toward our prefix. Therefore, we gathered a list of valid IP addresses of `http` servers based on a one-month Netflow trace [5]. We selected `http` server addresses because there were a wide variety of these inside the trace and we supposed that these addresses were persistent in time. We kept at most five addresses for each prefix in our BGP table. We completed this list of addresses by randomly selecting other IP addresses inside prefixes with less than 5 `http` servers inside the trace. We sent TCP `SYN` segments on port 80 to each address in the list, kept the addresses that answered and completed this new list with random addresses again. This way, we incrementally built a list of IP addresses responding on port 80 until we obtained at least one IP address responding for around 56000 prefixes out of the 125000 prefixes present in the BGP routing table. Moreover, the prefixes involved in our measurements belong to 75% of the ASs present in the Internet at the time the measurements were performed.

We chose to send TCP `SYN` messages on port 80 due to firewalls and Intrusion Detection Systems (IDS). Our first approach consisted in sending ICMP echo request messages. These were often blocked at firewalls or by IDS after a few trials. This black-listing did not occur as much on the TCP `SYN` messages sent toward `http` servers from our trace. However, we only performed one measurement each day in order to avoid this black-listing.

For each measurement, we modified the BGP configuration of the router in AS2111, restarted the BGP sessions with this new configuration and only started sending the TCP `SYN` segments two hours after the establishment of the BGP ses-

[5]The Netflow trace was collected at the Université catholique de Louvain (UcL) during a period from January 2003 till beginning of February 2003 (4th).

sions to ensure the convergence of BGP for our route before the measurement. For each measure, we sent exactly one TCP `SYN` segment, with a given sequence number[6], to each of these destinations. We captured the responses on each interface by means of the `tcpdump` tool. This way, we determined, for each measurement, the provider used by a given prefix to join our prefix.

2.2.2 AS-path Prepending Measurements

In this section, we look at the results obtained from the prepending measurements performed according to the methodology presented in section 2.2.1.

For our first measurement, we advertised our `/20` prefix without AS-Path prepending to our two providers. The first line of Table 2.2 shows that 67.82 % of the responses to the TCP `SYN` segments sent are received via Belnet. This difference is due to the variation in connectivity between our providers. Belnet is attached to two large ISPs and GEANT while Be_ISP is only attached to one large ISP. If we look at results at the AS-level, 64% of the responding ASs sent all their replies via Belnet while only 27% of the replies arrived via Be_ISP.

A second interesting result is that for 9% of the ASs, we received replies via our two providers. Those replies came from different prefixes belonging to these ASs. This can be explained by two factors. First, most large ISPs use hot-potato routing to route the transit traffic. Consider for example a router of a Tier-1 ISP that peers with AS3356, AS1299 and the European Commercial ISP shown in Figure 2.9. When this router receives a packet whose destination is inside AS2111, it will send the packet to the closest router that is connected to one of the providers of our providers. Another router from the same Tier-1 ISP may select another transit AS to reach AS2111. This explains why, in our measurements, 9 of the 20 Tier-1 ISPs and 92 stub ASs advertising two or more prefixes but connected to a single provider, according to [SARK02], sent replies via our two providers. These stubs are likely to be multiply connected to the same provider. However we do not see multiple links between two ASs in [SARK02]. This is due to the inference of the Internet topology based on BGP routing tables. Second, some stub ASs such as cable-modem providers are present in different cities. These ASs often use the cheapest provider in each city. They thus select different routes in different cities.

We then evaluated different amounts of AS-Path prepending via our two providers. Table 2.2 shows the impact of AS-Path prepending on each BGP session.

Table 2.2 shows that without prepending, the majority of prefixes reach AS2111 via Belnet. When we look at the prepending of the AS-Path for our route advertisement on Belnet session (Table 2.2), we note that prepending our ASN once is enough to reverse the initial situation. Around 80% of the prefixes now respond via the Belgian ISP. Prepending the AS-Path twice still increases the ASs using the Belgian ISP to join our prefix. Additionally, we see that prepending the AS-

[6]The sequence number is used to distinguish the responses to our segments from junk traffi c.

	Prepend to Belnet		Prepend to Be_ISP	
	Upstream		Upstream	
	Belnet (%)	Be_ISP (%)	Belnet (%)	Be_ISP (%)
no prepending	67.82	32.18	67.82	32.18
prepend once	22.22	77.78	79.64	20.36
prepend twice	15.67	84.33	80.87	19.13
prepend three times	15.35	84.65	100	0

Table 2.2: Impact of AS-Path prepending (prefix)

Path three times on Belnet session doesn't have a significant impact compared to the distribution obtained by prepending the AS-Path twice on this session. Other prepending measurements revealed that these prefixes could not be moved with an increased amount of prepending. It is very likely that local preferences influence the way their ASs send their traffic. For example, ASs connected to GEANT may favor GEANT for their outgoing traffic. This cannot be verified because it depends on the policies of distant ASs. However, as investigated later in this section, Figure 2.10 seems to confirm our assumption.

The same conclusions can be drawn for prepending on the link with the Belgian commercial ISP, in Table 2.2. Prepending the AS-path once on the Belgian ISP session is enough to receive around 80% of the responses on the Belnet link. There is not much difference between prepending the AS-path once and twice. This is due to the length of the AS-path in the advertisements received by the sources for our prefix. The selection of the best route towards AS2111 depends on the policies enforced by other ASs as well as on the location of the Belgian ISP in the hierarchy of ASs. In this case 20% of the sources still prefer the path through the Belgian ISP even after this path is prepended twice. This is probably due to local preferences enforced by the European commercial provider for the customer routes of the Belgian ISP. Therefore, we see that it is required to prepend the AS-Path three times on the link to the Belgian ISP in order to switch all incoming traffic to the Belnet link.

To better understand the impact of `AS-Path` prepending, we selected from the BGP routing tables of our two providers, the list of their 10 most important upstreams. Those upstreams are the transit ASs located at two AS-hops from our providers that advertise routes for a large number of prefixes from which we receive replies via this provider. The largest of those ASs are mainly large Tier-1 ISPs.

Figure 2.10 shows the number of prefixes that we can reach via those ASs for which the replies are received via Belnet. For example, without prepending, Belnet received replies from more than 5000 prefixes whose routes received by Belnet contained `ASXX:AS701:*` in their AS-Path (where `ASXX` matches one of Belnet's providers). Note that since the Internet paths are asymmetrical, the AS-Path found in the BGP routing table of Belnet indicates the path used to send packets toward a given prefix, not the path used by those prefixes to reach our AS. Figure 2.11 provides the same information for the ten most important upstreams of Be_ISP. We see in Figures 2.10 and 2.11, that, for example, packets from prefixes that are reachable via AS701 (UUnet) and AS1239 (Sprint) are received via both Belnet and Be_ISP when no prepending is used.

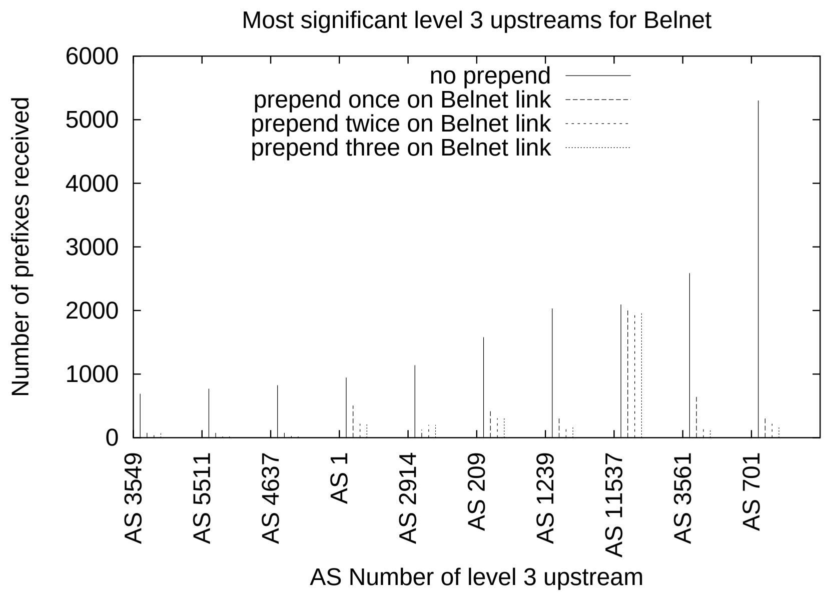


Figure 2.10: Important upstreams behind Belnet

When prepending is used on the Belnet link, Figure 2.10 shows that most of the prefixes move away from the Belnet link. However, a small fraction of the prefixes is not affected by the AS-Path prepending. AS11357 (Abilene) is special since almost none of the prefixes behind this AS move with AS-Path prepending. Abilene connects multiple universities and is connected to GEANT. For these universities, the Abilene connection is usually much cheaper than their commodity Internet connection and our measurements indicate that they prefer their Abilene connection whenever possible.

Figure 2.11 shows that most of the responses from prefixes behind the 2 major upstreams do not move after prepending our announce on the Belgian ISP. However, over 50% of the responses from prefixes behind AS3549 (Globalcrossing) are received through Belnet when prepending is performed. When prepending three times is used on the link to the Be_ISP, all replies are received via the Belnet link. Note that those measurements correspond to the replies received from 56.000 prefixes. Some prefixes, not responding to our TCP `SYN` segments, may still send packets via the Be_ISP link even with prepending three times. To validate the results obtained for our measurements, we used the BGP routing tables collected by RIPE [AU99] and Routeviews [Rou]. From the BGP tables collected at these sites,

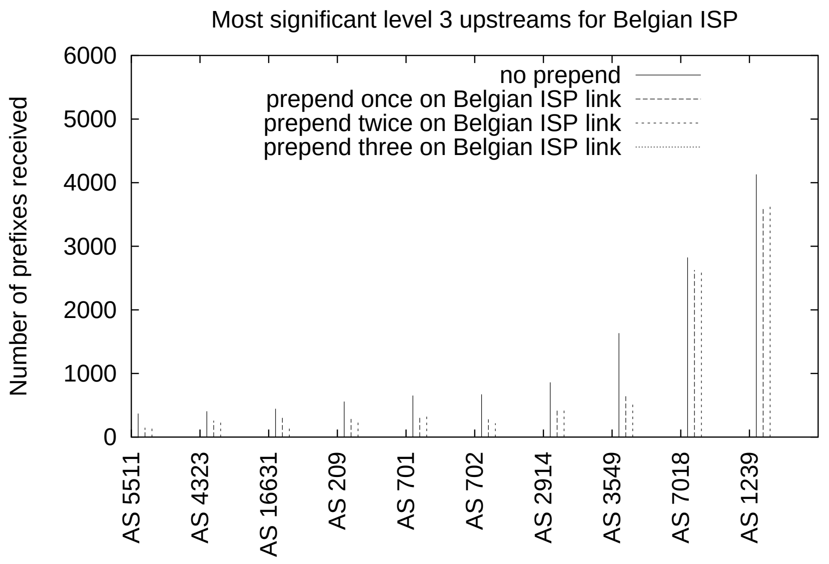


Figure 2.11: Important upstreams behind Be_ISP

we were able to corroborate our measurements. We also found one AS, AS11608 that did not respond to our TCP `SYN` segments and continued to use an AS-Path containing Be_ISP to reach AS2111 when the path via Be_ISP was prepended.

2.2.3 Community-based TE Measurements

Besides AS-Path prepending, another technique that is often used to control the flow of the incoming traffic is to rely on BGP communities as introduced in section 2.1.2. The actions of the communities defined by providers typically apply toward a large AS (e.g. tier-1 or tier-2 ISPs providing transit service), an interconnection point, a country or a continent. Unfortunately, all ISPs do not support all communities. For our measurements we had to rely on the communities supported by the providers of Belnet and Be_ISP.

Multiple community values can be attached to a route. However, in this section, we illustrate the influence of using single `do not announce` communities on the incoming traffic. In our measurements, we first attached to the advertisement of our prefix toward the Belgian commercial ISP a community preventing the redistribution of our route by the European provider of Be_ISP to Sprint (AS 1239), then to AT&T (AS 7018) and, finally, to Globalcrossing (AS 3549). These ASs are the three most important upstreams behind Be_ISP. The results of these three measurements are presented in Table 2.3. We note that a small portion of the ASs responding through Be_ISP reach our prefix through Belnet, when our prefix is not announced to Sprint by the European ISP. The same observation is made for the community preventing the redistribution of our route to AT&T by the European commercial ISP. However, the `do not announce` community toward Globalcrossing does not imply a move of the responses toward Belnet. Analogous results are obtained when using a community that requests the provider of Be_ISP to `prepend 3 times` its AS number when advertising our route to respectively AS1239, AS7018 and AS3549.

	Upstream		
	Belnet (%)	Be_ISP (%)	Both (%)
No communities	64	27	9
AS 1239	71	21	7
AS 7018	71	22	8
AS 3549	58	27	14

Table 2.3: Do not announce toward specified peer on the Be_ISP link

The BGP communities can be used to achieve a finer control than `AS-path` prepending on the incoming traffic. However, it should be noted that they suffer from three important drawbacks.

The first drawback is that, given our limited knowledge of the Internet topology and the routing policies used by distant ASs, it is difficult to predict the impact of a given community value. For example, consider Figure 2.9 and assume that AS2111 attaches to its route advertised to Belnet a community indicating that Belnet should not advertise the route to AS3356. In this case, AS3356 will not use its link with Belnet to reach AS2111. From Figure 2.9, AS3356 will send its packets to either AS1299 or the European Commercial ISP. In the first case, the community used by AS2111 does not have any effect on the packets received by AS2111. Furthermore, the sources that are downstream of AS3356 will recompute their best route to reach AS2111 and some of them may use AS1299 instead of AS3356 to reach AS2111 while others will utilize other paths. Given our limited knowledge of the Internet topology, it is very difficult to predict the decision that all those ASs will take.

A second drawback of the BGP communities is that the impact of one community on the incoming packet flow will depend on whether it is associated with other communities or not. For example, consider the topology of Figure 2.12. This topology is a subset of the AS-level topology inferred by [SARK02] on January 9th, 2003. Assume that AS17049 uses a community to request AS6467 to not announce its route to AS701. In this case, AS701 may update its BGP routes and use its peering link with AS1 to reach AS17049 via AS6467. Thus, the community has no effect on the packet flow as seen by AS17049. However, if this community is used together with a community requesting AS6467 to not advertise the route to AS1, then both AS701 and AS1 will probably use AS1239 to reach AS17049.

Finally, the last drawback of the utilization of communities is that a typical AS will need to choose among a large number of different communities. For example,

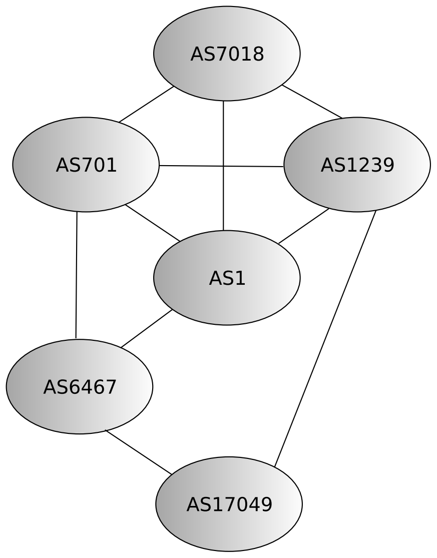


Figure 2.12: Example with multiple communities

consider the communities that allow a stub to influence the advertisement of its routes to the peers of its peers, as in our measurements. The number of available communities depends on the number of ASs that are two AS-hops away. For Belnet, there are 1729 distinct ASs at two AS-hops. In practice, however, it can be expected that such communities will mainly be used on customer-provider links. [QPBU05] has shown that, in the topology from [SARK02], twenty percents of the multi-homed stubs have more that ten peerings with providers at two hops. Thus, these stubs can use $2^{10} = 1024$ sets of communities to engineer their incoming traffic with communities influencing the redistribution of their route toward providers only. Additionally, sixty percent of stubs have more than 500 links at 2 hops. This implies that a lot of communities' combinations exist to engineering the traffic of these stubs even if we exclude the communities targeting single-homed customers since this traffic cannot be moved with communities.

2.2.4 Discussion on BGP TE

In this section, we explained why it is difficult for an Autonomous System to control the flow of its incoming traffic with BGP.

We have presented a detailed measurement study of AS-Path prepending. Our measurements clearly show that the granularity of AS-Path prepending is very limited. Moreover, [QPBU05] shows by means of simulations that the tie-breaking rules of the BGP decision process are responsible for the selection of 30-50% of the routes in the global Internet. This means that at a given router receiving multiple routes for the same prefix the chances that the routes have the same `AS-path` length are high. These simulations confirm the low granularity of the `AS-path` prepending technique. In practice, this technique is used to configure a backup link [QUP$^+$03]. The `AS-path` of routes advertised on the backup link is prepended to avoid carrying traffic on the backup link whenever possible. However, this technique is difficult to use in order to balance the incoming traffic.

The use of communities gives a finer control on the incoming traffic. However, we have shown with our measurements that the impact of a given community is unpredictable and the combination of multiple communities to achieve a given load distribution objective reveals itself to be hard.

To accurately control the flow of its incoming packets, an AS should be able to predict which route will be selected by distant ASs. Unfortunately, this prediction is difficult for two reasons. First, our knowledge of the Internet topology and the routing policies is incomplete. Second, even with a detailed topology, it would still be very difficult to predict the outcome of the tie-breaking rules of the BGP decision process.

Based on our analysis, the current BGP-based techniques are not appropriate to control the incoming packet flow. This is consistent with the opinion expressed by the authors of [ACE$^+$02]. They say that most methods manipulating the point at which inbound traffic enters a network from an eBGP peer (such as inconsistent route announcements between peering points, `AS-path` prepending, and sending MEDs) are either ineffective, or not accepted in the peering community. They add that these methods are usually applied in a trial-and-error fashion and that a systematic approach for inter-domain traffic engineering is yet to be devised.

2.3 Related Work

Alternatives to BGP TE have been proposed. For example, [ACK03] proposes changes to the Internet architecture by using an overlay to BGP to transmit control information. This overlay is called OPCA for Overlay Policy Control Architecture. The objective is to be able to control the flow of the incoming traffic of an AS by negotiating the selection of inter-domain paths with remote ASs. The authors also consider the problem of reducing fail-over time in case of failure of an inter-domain path. The solution proposed in [QB05] also relies on cooperation between ASs. The source and destination ASs of the traffic cooperate in the establishment of IP tunnels to balance the incoming traffic of the destination AS or reduce the latency to the destination AS. The cooperation is performed by using eBGP multi-hop sessions.

[ABKM01] proposes a Resilient Overlay Network (RON) for data traffic. This overlay is built by the end-systems. The aim is to discover paths that are not available with BGP. Measurements are performed between end-systems and are used to determine the quality of the paths. Routing and packet forwarding take place at the application level.

[dBL03] relies on IPv6 to perform traffic engineering. Host in multi-homed sites are affected an address by each provider. The address used by an host determines the provider that is used to communicate with the host. In this proposal, hosts contact a server inside their domain to determine the source address to be used before starting a communication. The authors use this server to load-balance the outbound traffic. In [dUB05a], they go one step further. They propose to also select the destination address, among the addresses available for a host, based on delay estimations in order to use low latency paths.

In [YSLMB$^+$05] and [FBR$^+$04], the authors define an architecture with a centralized entity inside each domain. They propose to define a new interdomain routing protocol to be used between the entities, either called Inter-Domain Routing Agent or Routing Control Platform. They propose to exchange QoS information with this routing protocol. A mechanism for the negotiation of Service Level Specification (SLS) is also defined in [H$^+$05] as a support for the provision of QoS-based services.

2.4 Conclusion

In this chapter we provided an overview of the current interdomain TE techniques and proposals. We described the use of the BGP attributes to engineer the outgoing and the incoming traffic. We said that it was easier to engineer the outgoing than the incoming traffic because the way traffic leaves an ASs relies on local decision. On the contrary, the incoming traffic is impacted by TE decisions and policies at other ASs.

We evaluated two incoming TE techniques: `AS-path` prepending and BGP `communities`. Our evaluation was performed by means of active measurements. We registered an AS Number (ASN) for this purpose. This AS was a dual-homed stub. We collected a set of valid IP addresses for `http` servers. We advertised a prefix with BGP for this AS. We generated small amounts traffic from these servers. We measured the peering link used by these servers to reach our prefix for each BGP configuration. These servers belonged to 45% of the prefixes present in BGP routing tables. The prefixes involved in our measurements belonged to 75% of the ASs present in the Internet at the time the measurements were performed.

We have shown that `AS-path` prepending is too coarse. It does not allow a precise control on the traffic. In addition, our measurements have shown that the BGP communities allow a finer control on the incoming traffic. However, the impact of a given community on the incoming traffic cannot be predicted before its application. Thus, administrators have no other choice than applying them on a trial-and-error basis. Moreover, there exists a large number of combinations for the communities. Thus, it is not straightforward to use the BGP communities to achieve a certain TE objective. We conclude that BGP TE techniques are not sufficient for a precise control on the interdomain traffic.

Chapter 3

RSVP-TE Extensions to Support Interdomain LSPs

We have shown in chapter 2 that BGP does not provide a fine granularity to engineer the traffic entering an AS. In addition, we have seen in chapter 1 that TE also consists in the provision of Quality of Service (QoS) to the end-user flows. However, there is no QoS information associated with the BGP routes, today. Moreover, the selection of the best BGP route does not depend on the quality of the path in terms of delay, bandwidth, . . . [HFP$^+$02]. Thus, the best BGP route may not be suitable for a particular type of traffic with given QoS requirements. Several QoS extensions to BGP have been proposed in the literature [Jac03, Bou05]. However, these extensions are far from being deployed. Furthermore, adding QoS information to BGP route advertisements may lead to a large increase in the number of BGP update messages exchanged in the Internet, if this QoS information is subject to frequent changes. It may lead to a large increase in the size of the update messages and BGP routing tables if the services require various QoS properties to be advertised.

Because MPLS enables to bypass classical IP forwarding, QoS may be provided to flows by establishing MPLS LSPs with RSVP-TE. These LSPs do not necessarily follow the routes distributed by BGP and installed in IP routers. Moreover, resources can be reserved for the LSP along its path. The problem of finding an interdomain path that respects certain QoS constraints will be dealt later in this document.

Traffic engineering also aims at the robustness of the network in case of failures. Currently, when a failure occurs on an interdomain link, the BGP router that detects the failure withdraws the route advertisements corresponding to destinations reachable through the failed link. This withdrawal causes other BGP routers to remove the corresponding routes from their routing table and to try to find a better route. Due to the way the best route is selected by the BGP decision process and announced by BGP, a BGP router may announce and withdraw successively several distinct routes for the failed destination until the best route according to the policies is found. Combined with the utilization of various timers in the BGP implementations to avoid generating bursts of withdraw and advertisement messages, this can lead to long restoration times in the Internet. An experimental study of the BGP convergence in the global Internet [LABJ00] has shown that the restoration time for a route was on the order of a few minutes with an average fail-over delay of about 3 minutes. During that time there is a loss of connectivity from several sources to the destination prefix of the route. Such a delay is far too long for interactive and transaction-oriented network applications. Several studies have shown that topology factors and changes to BGP implementations could improve the BGP convergence. Moreover solutions, such as EPIC [CDZK05], an enhanced path vector algorithm, to limit path exploration in BGP are also proposed. However, it can be expected that even in these conditions, a path-vector protocol relying on policies like BGP will have a restoration time that is one order of magnitude larger that the restoration time of intradomain routing protocols.

The use of MPLS across AS boundaries could be an efficient solution to provide shorter restoration times than those obtained with the Border Gateway Protocol (BGP) in case of inter-domain failures. In case intradomain LSPs are locally protected by means of pre-established LSPs, as seen in chapter 1, failure recovery is below 100 ms. It consists in the local failure detection and the switching of the traffic onto a pre-established backup LSP. Such restoration time can also be achieved for locally protected interdomain LSPs. In this chapter, we provide a way to pre-establish backup LSPs protecting interdomain resources.

Finally, besides the utilization of MPLS across AS boundaries for TE purposes, MPLS could be an efficient solution to support inter-AS VPNs [RR06] and to build more scalable Internet eXchange (IX) points [NEN02].

For the above reasons, we suggest, in this thesis, the use of MPLS with RSVP-TE to engineer the interdomain traffic. In this chapter, we describe the TE extensions to RSVP, proposed in [ABG$^+$01], for intradomain LSPs. Then, we present major requirements that have been formulated by the IETF for interdomain MPLS TE. Finally, we provide extensions to enable the establishment of traffic engineered interdomain LSPs toward a prefix or an AS destination and the local or global protection of these interdomain LSPs. The challenge is to provide mechanisms that ensure the confidentiality of the intradomain resources while being able to protect interdomain LSPs. Our contributions to the establishment and protection of interdomain LSPs have been published in [PB03, PB02] and [CP04]. [PB02] was the first publication on inter-AS RSVP-TE.

3.1 Intra-AS MPLS Terminology

In this section, we introduce the terminology used in this chapter to describe the establishment and the protection of intra-AS LSPs. This terminology is inspired from [ABG$^+$01], [SH03] and [PSA05]. Part (a) of figure 3.1 illustrates the roles of the different routers participating in the establishment of a protected LSP and its end-to-end disjoint LSP.

A Label Switching Router (LSR) is a router that may be crossed by MPLS Label Switched Paths (LSPs).

The head-end LSR is the router that initiates an LSP and is the source of this LSP. $R11$ is an head-end LSR in figure 3.1.

The tail-end LSR is the last router on the path of an LSP; it is the destination of this LSP. $R31$ is the tail-end router in figure 3.1.

The path of an LSP is the list of links and nodes crossed by the LSP.

The upstream link of a node is the link directly connected to the node and used to join this node on the path of an LSP.

The downstream link of a node is the link directly connected to the node and used to leave this node on the path of an LSP.

The upstream node of a node is the node directly upstream of the current node on the path of the LSP.

The downstream node of a node is the node directly downstream of the current node on the path of the LSP.

A Shared Risk Link Group (SRLG) is a group of links that may fail at the same time. An example of SRLG is provided later, in section 3.7.2. It is a set of links that share a common physical resource such as an Ethernet switch, a fiber, an optical cross-connect, ...

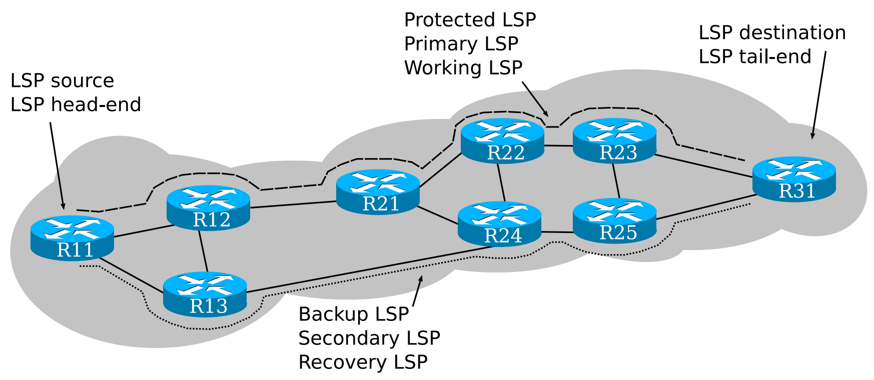
The protected LSP/primary LSP/working LSP is the LSP that is to be protected. It carries traffic before a failure occurs. The protected LSP follows path $R11 - R12 - R21 - R22 - R23 - R31$ in figure 3.1.

The backup LSP/secondary LSP/recovery LSP is the LSP that is used to restore traffic after the occurrence of a failure. In figure 3.1, the backup LSP is end-to-end disjoint from the protected LSP. It follows path $R11 - R13 - R24 - R25 - R31$.

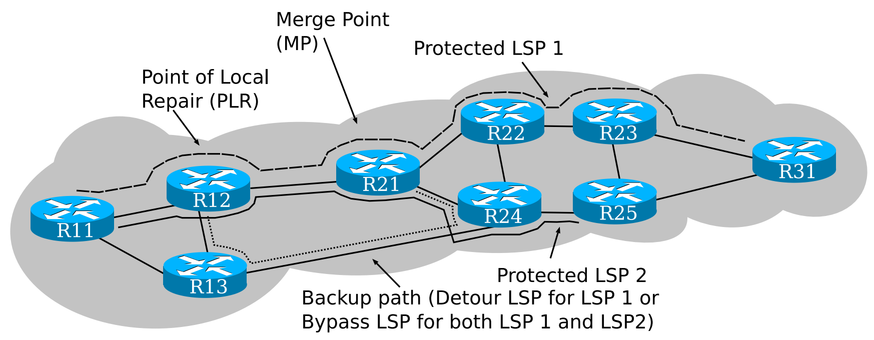
Now we define the notions related to the local protection of an LSP. Part (b) of figure 3.1 illustrates these concepts.

Local repair/protection is the technique to recover from a neighboring link and/or node failure.

A detour LSP is an LSP that locally protects a single protected LSP. The LSP with path $R12 - R13 - R24 - R21$ is a detour LSP that protects $LSP1$ against the failure of link $R12 - R21$, in figure 3.1 (b).



(a)



(b)

Figure 3.1: Intra-AS MPLS terminology

Backup path is the generic term for detour LSP and bypass tunnels.

A bypass LSP/tunnel is an LSP that locally protects a set of LSPs passing over a common facility. The protected LSPs and the bypass tunnel all share the same Point of Local Repair and Merge Point. In figure 3.1 (b), the backup path is a bypass tunnel if it protects LSPs 1 and 2 against the failure of link $R12 - R21$.

The Point of Local Repair (PLR)/ the path switch LSR (PSL) is the head-end of the backup path. $R12$ is a PLR in figure 3.1 (b).

The Merge Point (MP)/the Path Merge LSR (PML) is the node, where merging of the backup path with the protected LSP occurs. The MP for the backup path in figure 3.1 (b) is router $R21$.

3.2 Intra-AS LSPs

The specification of RSVP-TE [ABG$^+$01] defines extensions to the Resource reSerVation Protocol (RSVP) in order to establish traffic engineered LSPs. Among these extensions are the ability to distribute labels and to specify a strict or a loose path to be followed by an LSP.

RSVP messages are composed of a header followed by a sequence of objects. Among these objects are the Session Object, the Sender Template Object, the Explicit Route Object (ERO) and the Record Route Object (RRO). The Session and the Sender Template Objects are used to identify an LSP[PSA05] [1]. Based on the values stored inside these objects, a router is able to create and/or access the path-state and resv-state related to this LSP.

In an AS composed of a single area, the head-end LSR of an intradomain LSP is able to compute the complete strict path of the LSP by means of the Constrained Shortest Path First (CSPF) algorithm. It stores the nodes along this path in the ERO object. When an AS is divided into areas, the head-end LSR may compute with CSPF a strict path to the egress Area Border Router (ABR), followed by a loose path.

The routing of RSVP-TE `Path` messages is performed on the basis of the ERO, when it is present. This object contains a list of subobjects representing abstract nodes to be crossed by the LSP. Abstract nodes may either be a single node or a group of nodes such as a network prefix or even an entire AS. Subobjects inside the ERO can be marked with a “loose bit” to indicate that the subobject may be reached after crossing nodes that are not present inside the ERO[2]. Intermediate LSRs may complete the ERO when they meet an abstract node or a node marked with the “loose bit” inside the ERO. When no ERO is present inside a `Path` message, it is routed as a normal IP packet based on the packet’s destination, i.e. the IP Destination Address (IPDA) or based on the destination mentioned in the Session Object.

The path of an LSP can be recorded by using the RRO. This object is inserted inside `Path` and `Resv` messages by their source. Each LSR crossed by such message adds its address inside the RRO and stores the RRO inside the LSP’s path-state or resv-state. By inserting the RRO both inside `Path` and `Resv` messages, each LSR on the path of the LSP can obtain the complete path of the LSP. This information is useful for loop detection, route pinning, label recording and for the computation of disjoint LSPs.

Figure 3.2 illustrates the establishment of an LSP inside an AS composed of a single area. The head-end LSR $R0$ sends an RSVP Path message inside an IP packet with IPDA set to $R8$. The ERO of the Path message is a list of strict nodes. It is obtained from a CSPF computation at $R0$. The path computed with CSPF at

[1] In this document, we consider that an LSP is identifi ed according to the sender template-specifi c method. This method and the path-specifi c identifi cation method are defi ned in [PSA05]

[2] When the “loose bit” is not set, the `Path` message has to reach the following node inside the ERO without crossing intermediate nodes.

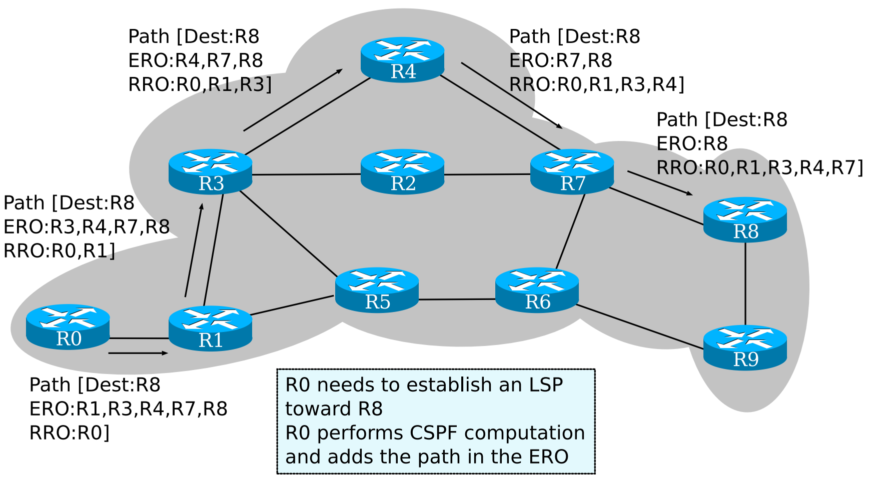


Figure 3.2: Establishment of an intradomain LSP

$R0$ toward $R8$ is $R0 - R1 - R3 - R4 - R7 - R8$, assuming all links have the same metric value and sufficient available bandwidth to support the LSP. Upon reception of the Path message, each node checks if the first node inside the ERO is one of its address. In this case, it removes the first node from the ERO and sends the Path message to the next node in the ERO, because the next node is a strict node and it is directly connected to the current node. The behavior of an LSR dealing with an ERO composed of abstract and loose nodes is described later, in section 3.6.2. $R0$ includes an RRO with its address inside the Path message. Each node that receives the message adds its address inside the RRO. Thus, each node learns the upstream path followed by the LSP from the RRO. And, the tail-end, $R8$, knows the complete path from the RRO. $R8$ inserts an RRO with its address inside the Resv message that is sent back to $R0$. The Resv message is sent along the path of the LSP. Each node that receives this message adds its address inside the RRO. Consequently, the nodes crossed by the LSP learn the downstream path of the LSP from the RRO of the received Resv message and the head-end LSR, $R0$, learns the complete path from this RRO.

3.3 Protection of Intra-AS LSPs

There are different ways to protect intra-AS LSPs [SH03]. With “global repair”, a working LSP is end-to-end or segment protected with a recovery LSP, link/node disjoint from the protected portion of the working LSP. An alternative is to individually protect links and/or nodes of the working LSP with a recovery LSP. This is called “local repair”.

In case of failure, a message notifying the failure has to travel all the way back to the head-end LSR when the LSP is end-to-end protected [3]. On the other hand, with local repair, the notification message only travels backward to the Point of Local Repair (PLR), that is the head-end of the backup LSP. This router is closer to the point of failure allowing faster restoration times than with global repair.

Local repair can be provided by Detour LSPs or Bypass Tunnels [PSA05]. These terms are defined in section 3.1. In this document, we use "backup path" or "backup"as a generic term for Detour LSPs and Bypass Tunnels. A Next-Next-Hop (NNHOP) Detour LSP protects a single LSP against a node failure and against the failure of the link used by the LSP to join that node, i.e. its upstream link. A Bypass Tunnel however protects a set of LSPs passing over a common facility [PSA05].

In the local protection of an intra-domain intra-area LSP, the path of a Detour LSP or Bypass Tunnel may be completely computed by the PLR. The CSPF algorithm may be used for this purpose. In this algorithm, first, the resources to be avoided by the backup LSP are pruned from the topology, then a Shortest Path First (SPF) computation is performed.

In order to locally protect LSPs against single link and node failures, two objects are defined: the FAST_REROUTE Object and the DETOUR Object. The FAST_REROUTE Object is carried inside the `Path` message of the primary LSP. Coupled with the SESSION_ATTRIBUTE Object, it indicates the type of protection required for the primary LSP. It also provides the constraints for the path of the backup. The Detour Object is carried inside the `Path` message of the backup, i.e. the LSP protecting a segment of the primary LSP, in the Detour mode[4]. The Detour Object contains the address of the PLR and of the node to be avoided. Finally, flags are defined for the RRO sub-objects to indicate whether protection is provided, the type of protection provided, and whether protection is in use.

The node, where merging of the backup path with the primary LSP occurs, is called the "Merge Point (MP)". This LSR may be any router on the path of the primary LSP downstream from the PLR and the resource to protect.

3.4 Inter-AS LSP Requirements

In [RV05], several requirements for MPLS inter-AS Traffic Engineering (TE) are expressed. Among these requirements is the desire of SPs to keep internal AS resources and the set of hops followed by the TE-LSP confidential. This **confidentiality requirement** implies the capability of partly specifying the hops that the TE-LSP must traverse since global topology information is not available. Moreover, it must be possible to perform **path optimization** inside each transited AS, where the required information is available. In addition, end-to-end optimization

[3]A solution called Bidirectional Forwarding Detection (BFD) is at the early stage of development at the IETF, in the BFD Working Group. Its functioning for the detection of failures along MPLS LSPs is described in [AKNS06]. The objective of this solution is to achieve sub-second data plane failure detection.

[4]Bypass tunnels are not signalled with the Detour object.

of inter-AS LSPs is also required by SPs.

A second requirement concerns the restoration capabilities of inter-AS LSPs. The proposed solution has to be able to provide rapid **local protection** against link, Shared Risk Link Group (SRLG)[5] and node failures. Additionally, it should support the establishment of multiple link/SRLG/node **diversely routed inter-AS TE LSPs** between a pair of Label Switching Routers (LSRs).

A last requirement is that the proposed solution should be **scalable** in terms of the amount of IGP flooding, the additional information carried by BGP, the amount of RSVP TE signaling messages exchanged and state to retain.

3.5 Inter-AS MPLS Terminology

In this section, we introduce the terminology related to the establishment and the protection of inter-AS LSPs. The terminology provided in section 3.1 is also used for interdomain LSPs. The additional terms provided in this section and their definitions are inspired from [CP04]. Part (a) of figure 3.3 illustrates the roles of the different routers participating in the establishment of a protected LSP and its end-to-end disjoint LSP. In figure 3.3, $R11$ is the head-end LSR and $R31$ is the tail-end router. The path of the protected LSP is $R11 - R12 - R21 - R22 - R23 - R31$. The backup LSP crosses nodes $R11$, $R13$, $R24$, $R25$ and $R31$.

The AS path of an LSP is the list of ASs that are crossed by the LSP.

The upstream AS ASx of an AS ASy directly precedes ASy in the AS path of an LSP. For example, $AS1$ is the upstream AS of $AS2$ along the protected and backup LSPs, in figure 3.3.

The downstream AS ASy of an AS ASx directly follows ASx in the AS path of an LSP. $AS2$ is the downstream AS of $AS1$ and $AS3$ is the downstream AS of $AS2$ along both the protected and backup LSPs, in figure 3.3.

An AS Border Router (ASBR) is a router that is at the boundary of an Autonomous System (AS).

The (primary) ingress ASBR/entry ASBR of an LSP is the first router crossed by the LSP when entering an AS. This term is mostly used for protected LSPs. $R21$ and $R31$ are ingress ASBRs.

The secondary ingress ASBR is the first router crossed by a backup LSP when entering an AS. $R24$ is a secondary ingress ASBRs.

The (primary) egress ASBR/exit ASBR of an LSP is the last router crossed by the LSP when leaving the AS. This term is mostly used for protected LSPs. $R12$ and $R23$ are egress ASBRs.

[5]An SRLG identifi es a set of links that may fail together. If one of the links belonging to an SRLG fails, all the other links belonging to the same SRLG may also be impacted by the failure.

The secondary egress ASBR is the last router crossed by a backup LSP when leaving an AS. $R13$ and $R25$ are secondary egress ASBRs.

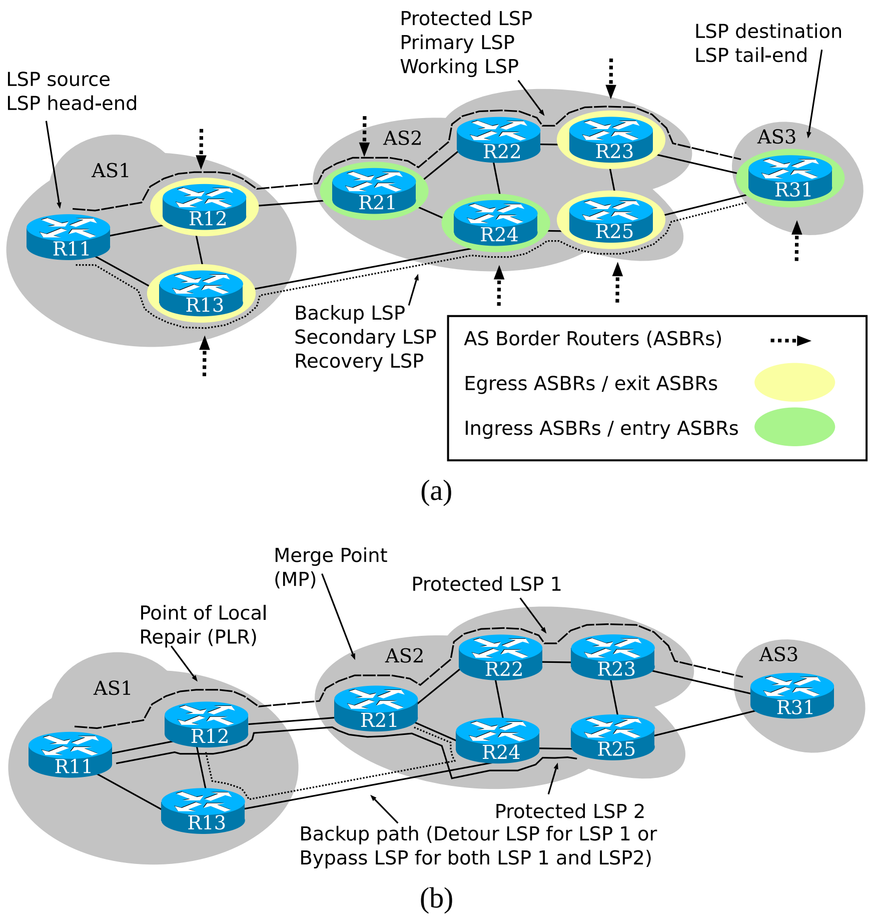


Figure 3.3: Inter-AS MPLS terminology

Now, we define the notions specifically related to the local protection of an inter-AS LSP. Part (b) of figure 3.3 illustrates these concepts. The other concepts are taken from the local protection of intra-AS LSPs. Their definition is provided in section 3.1. The LSP with path $R12 - R13 - R24 - R21$ is a detour LSP that protects $LSP1$ against the failure of link $R12 - R21$, in figure 3.3 (b). We have seen that "backup path" is the generic term for detour LSPs and bypass tunnels. In figure 3.3 (b), the backup path is a bypass tunnel if it protects LSPs 1 and 2 against the failure of link $R12 - R21$. $R12$ is a PLR in figure 3.3 (b). The MP for the backup path in figure 3.3 (b) is router $R21$.

Inter-AS link protection is the protection of an LSP against the failure of one of its inter-AS links. In figure 3.3, the backup LSP protects $LSP1$ against the failure of inter-AS link $R12 - R21$.

Inter-AS node protection is the protection of an LSP against the failure of one of its ASBRs.

Inter-AS SRLG protection is the protection of an LSP against a simultaneous failure of all links that belong to the SRLGs which contain the inter-AS link.

3.6 Inter-AS LSPs

In this section, we describe our solution to establish inter-AS LSPs. We introduce the RSVP-TE extensions required to enforce the requirements presented in section 3.4. Subsequently to [PB02], three different methods have been proposed at the IETF for the establishment of inter-AS LSPs. These methods are called contiguous LSPs [AV06a] , LSP stitching [AV06b] and LSP nesting [KR05]. A contiguous Inter-AS LSP consists of a single LSP that crosses multiple ASs [AV06a]. With LSP stitching, intra-AS LSPs are stitched at the border of the ASs to form an inter-AS LSP [AV06b]. LSP nesting [KR05] describes a solution where an inter-AS LSP is tunneled in intra-AS LSPs inside each AS. This section concerns the establishment of contiguous LSPs. We consider both contiguous and nested LSPs in section 3.7.

The desire of SPs to hide their internal topology, as currently achieved by BGP and the need for LSP's protection are not easily satisfiable simultaneously. Indeed, it is necessary for an LSR to know the path of an LSP to be able to protect it. This information is easily obtained from RSVP-TE objects for the intra-AS path of an LSP (see 3.2) but it is not so obvious to obtain such information for its inter-AS path when the confidentiality requirement regarding internal AS's topologies is observed. In this section, we propose extensions to RSVP-TE that fulfill both the confidentiality and the protection requirements concurrently while trying to keep our solution scalable. Our solution also tries to only impact the head-end LSR, the intermediate AS Border Routers (ASBRs) on the path of the inter-AS LSP and the tail-end LSR of the LSP therefore allowing a smooth migration toward the support of inter-AS LSPs. Our solution does not impact the current BGP and MPLS Traffic Engineering techniques. Moreover, it does not require additional IGP flooding. And last but not least, our solution supports the dynamic establishment of inter-AS LSPs avoiding the need for static configuration at the head-end LSR of the inter-AS LSP.

3.6.1 Destination of an LSP

The first problem encountered in the dynamic establishment of inter-AS LSPs for TE purposes is that unless the head-end LSR has been manually configured with the IP address of the tail-end LSR, the head-end LSR cannot know this information. The head-end LSR knows the prefix destination or the AS destination of the traffic to engineer. The IP address of the LSP's tail-end cannot be determined by the head-end LSP, before establishing the tunnel, on the basis of its BGP routing table. The

BGP routing table contains only information about destination prefixes and their AS paths. It does not provide valid IP addresses inside these prefixes.

To solve this problem, we propose to enable the establishment of LSPs based on a prefix or on an AS number and a prefix destination. During the establishment of an LSP based on a prefix destination, the `Path` message is forwarded through the network until it reaches an LSR with an IP address that belongs to this prefix . The `Path` message itself is routed on the basis of its destination IP prefix and possibly along an explicit route defined by an Explicit Route Object (ERO). The second type of destination that we propose is composed of an AS number and an IP prefix. In this case, the `Path` message is forwarded through the network on the basis of the destination prefix until it reaches an LSR that is part of the specified AS independently of the destination prefix. The path followed by the `Path` message can also optionally be specified with an ERO object. It is necessary to specify a prefix in addition to the AS number because BGP only provides prefix based routing.

These AS+prefix or prefix destination types are necessary to send the first `Path` message. However, once the first `Resv` message has been received, the source LSR of the LSP knows the IP address of the destination LSR. But, since the identification of an LSP is composed of the destination of this LSP. It is not desirable to change this destination once the LSP has been established and, therefore, the same destination is used for consecutive `Path` refresh messages.

The modifications to RSVP-TE that we propose to support AS+prefix and prefix destinations are described in appendix B of [PB02]. We propose to define two flags for the IPv4 and IPv6 prefix ERO subobjects. These flags indicate if the prefix is used for routing purposes, in case of an AS+prefix destination, or if the prefix is a loose destination, in case of prefix destination.

3.6.2 Explicit Routing of an LSP

The Explicit Route Object (ERO) is well suited for the establishment of inter-AS LSPs in that it enables the head-end of the LSP to partially compute the path to be followed by the LSP. Following nodes crossed by the `Path` message are able to complete this object as the `Path` message goes along. More precisely, the head-end LSR is only able to fill the ERO with nodes that belong to the same AS and eventually with the ASs that will be crossed by the `Path` message. At the entrance of each AS, the ASBR computes the path of the LSP toward the downstream AS and completes the ERO accordingly. This is illustrated in figure 3.4 where $R0$ computes the path toward $AS1$ and sets the ERO accordingly. Inside $AS1$, $R3$ completes the ERO toward the next AS, $AS3$ and so on. These paths are computed based on the destination prefix: $65.0.0.0/8$.

The ERO object may be constructed at the head-end LSR either based on a manual configuration that specifies the ASs and/or the ASBRs to be crossed by the LSP, based on the BGP routing table, based on a Path Computation Element

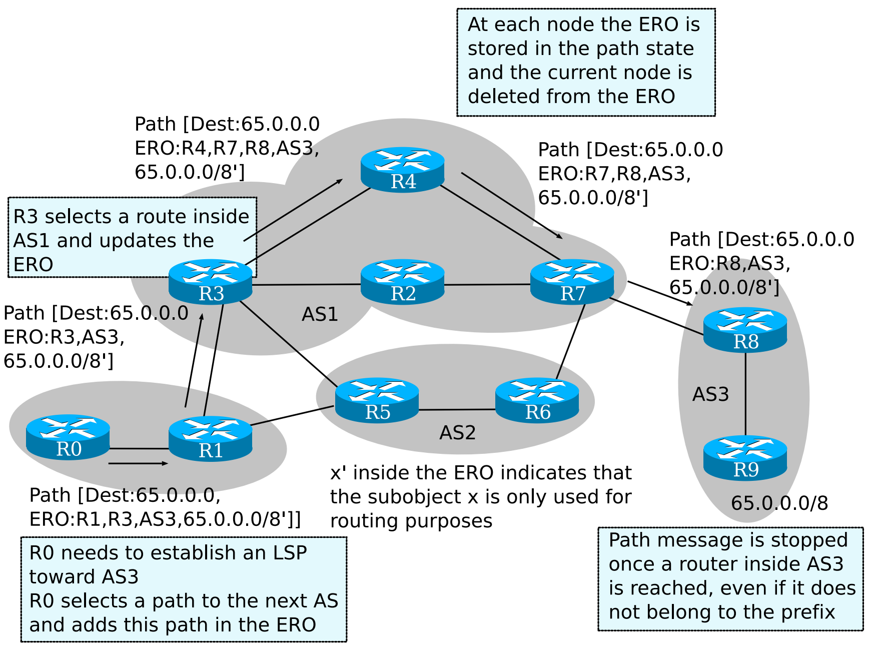

At each node the ERO is stored in the path state and the current node is deleted from the ERO
Path [Dest:65.0.0.0 ERO:R4,R7,R8,AS3, 65.0.0.0/8']
Path [Dest:65.0.0.0 ERO:R7,R8,AS3, 65.0.0.0/8']
R3 selects a route inside AS1 and updates the ERO
Path [Dest:65.0.0.0 ERO:R8,AS3, 65.0.0.0/8']
R4
R3
R2
R7
R8
AS1
Path [Dest:65.0.0.0 ERO:R3,AS3, 65.0.0.0/8']
AS3
R5
R6
R0
R1
AS2
R9
x' inside the ERO indicates that the subobject x is only used for routing purposes
65.0.0.0/8
Path [Dest:65.0.0.0, ERO:R1,R3,AS3,65.0.0.0/8']]
R0 needs to establish an LSP toward AS3
R0 selects a path to the next AS and adds this path in the ERO
Path message is stopped once a router inside AS3 is reached, even if it does not belong to the prefix

Figure 3.4: Establishment of an inter-AS LSP

(PCE)[6] or based on a solution to be provided by the `IPsphere FORUM` [IPs]. The inter-domain path selection could be performed by relying on QoS information distributed by extensions to BGP proposed in [XLWN02], [CJ03] and [PQB03]. However, we do not believe that QoS extensions to BGP will scale in the Internet. Later in this document, we evaluate other inter-domain path selection techniques. The ERO specifies only a set of hops on the path of the inter-AS LSP and it leaves to each crossed AS the responsibility of the local path optimization according to a set of constraints also carried inside the `Path` message of the LSP[7]. This fulfills the local path optimization requirement from the first paragraph of section 3.4.

3.6.3 RRO Aggregation

The Record Route Object (RRO) enables to obtain the path followed by an LSP leading to its usefulness in detecting loops inside the LSP's path, the capability to pin the LSP onto its path and the possibility to compute LSPs disjoint from this LSP for global or local protection.

[6]A PCE is a path computation tool with whom LSRs may communicate with Path Computation request (PCReq) and reply (PCRep) messages as defined in [VLA+06]. The architecture for PCE-based path computation is described in [FVA06]

[7]LSPs' constraints are carried inside the Session Attribute Object [ABG+01], FAST_REROUTE object [PSA05], eXclude Route Object (XRO) [LFC05] and eXclude Route Sub-Object (EXRS) [LFC05].

We note that recording the path of an inter-AS LSP may be in contradiction to the SPs desire to hide the internal topology of ASs. Therefore, we propose to modify the processing of this object at the ASBRs so as to withhold from neighboring ASs the complete path followed by the LSP inside the current AS. We call this process "RRO aggregation".

The aggregation of the RRO consists in marking the subobject added by the ingress ASBR inside the AS. And, at the last router of the AS, i.e. the egress ASBR, the subobjects starting from the marked subobject, added by the nodes inside the AS, are removed. These subobjects are replaced by the address of the ingress ASBR, the AS number and the address of the egress ASBR in order to keep enough information to perform loop detection, disjoint path computation and route pinning of the inter-AS LSP. Figure 3.5 illustrates the aggregation of the RRO. In this figure, $R3$ adds its address inside the RRO and marks it. The following LSRs ($R4$ and $R7$) add their address inside the RRO. The egress ASBR, $R7$ in figure 3.5, removes all addresses starting from the marked subobject, representing the address of $R3$. It replaces these subobjects by the address of the ingress ASBR ($R3$), its AS number ($AS1$) and its own address ($R7$).

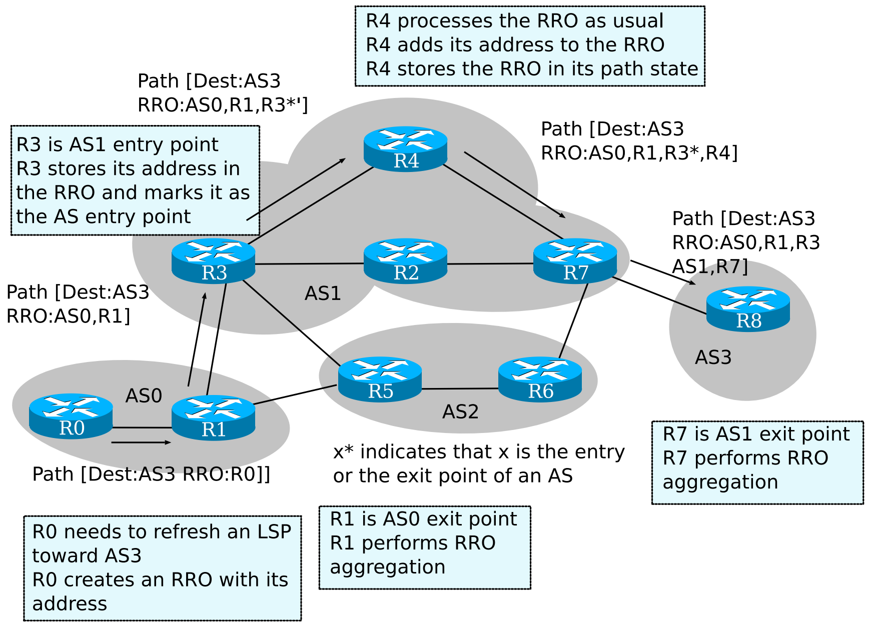


Figure 3.5: Processing of the RRO object

The modification of the RRO processing that we propose, only takes place at ASBRs and gives the opportunity to hide the internal topologies of ASs while still enabling to protect the established inter-AS LSPs, in conformance to the SPs' requirements.

In the following sections, we look at the establishment of LSPs that are totally or segment disjoint from an existing inter-AS LSP. Our first objective is to provide restoration capabilities analogous to the ones provided to intra-AS LSPs including local protection against link, node and SRLG failures. Further, the possibility to establish completely link or node disjoint LSPs can be useful to balance traffic on these disjoint LSPs and may be used for end-to-end protection.

3.7 Local Protection of Inter-AS LSPs

Among the different types of protection that may be provided to an inter-AS LSP we favor local protection over end-to-end protection of LSPs in order to leave to each operator flexibility in the choice of their preferred protection policy and to achieve faster traffic recovery. In this section, we propose mechanisms to provide local protection of links between 2 ASs, their SRLGs and of the nodes at the border of an AS. Techniques to protect AS core nodes and links joining these nodes are described in [PSA05].

In section 3.7.1, we propose to tunnel inter-AS LSPs through intra-AS LSPs inside an AS, as described in [KR05], as an alternative to RRO aggregation. This tunneling favors the confidentiality requirement concerning intra-AS topologies [RV05] as well as the establishment of inter-AS LSPs. The establishment of inter-AS LSPs will be studied in subsequent chapters. In this document, it is assumed that ASs define their SRLGs independently from the SRLGs in other ASs.

In section 3.7.2, we show that an end-to-end recovery LSP, crossing multiple ASs, can only provide link and node protection. For SRLG protection and fast recovery, the methods in [PSA05] have to be used. Section 3.7.3 and section 3.7.4 describe how these methods can be used for the protection of inter-AS LSPs with detour LSPs and bypass tunnels. Nodes other than those mentioned in this document must use the methods in [PSA05] to establish detour LSPs or bypass tunnels. Moreover, these nodes establish detour LSPs that merge with the working LSP in the same AS where they are originated, or these nodes establish/use bypass tunnels that terminate in the same AS as where they originate.

3.7.1 Inter-AS LSP Tunneled through an Intra-AS LSP

To enforce the confidentiality requirement of [RV05] we proposed, in section 3.6.3, to record an aggregate of the LSP's path in the RRO. With this technique, the RRO contains the necessary information for the computation of disjoint LSP segments. It informs each LSR, on the path of the LSP, about the ASs, the ingress and the egress ASBRs crossed by the LSP in addition to the complete path of the LSP inside the AS.

An alternative to RRO aggregation is to tunnel the inter-AS LSPs inside intra-AS LSPs [KR05]. The intra-AS path followed by these tunneled LSPs, inside each AS, is hidden outside the ASs. Thus, the confidentiality requirement is respected. In addition, this solution is more scalable. Core routers do not see the inter-AS

LSPs. They only have to support the intra-AS tunnels. However, the PLR for the failure of the exit ASBR is not the upstream LSR along the tunnel; it is the ingress ASBR. This node is farther away from the point of failure than the direct upstream LSR. Consequently recovery time is longer than with contiguous LSPs and RRO aggregation.

The procedures described in the following sections apply for inter-AS link, node and SRLG protection of inter-AS LSPs whether they are tunneled or not.

3.7.2 Problems in SRLG Protection with Disjoint End-to-end LSPs

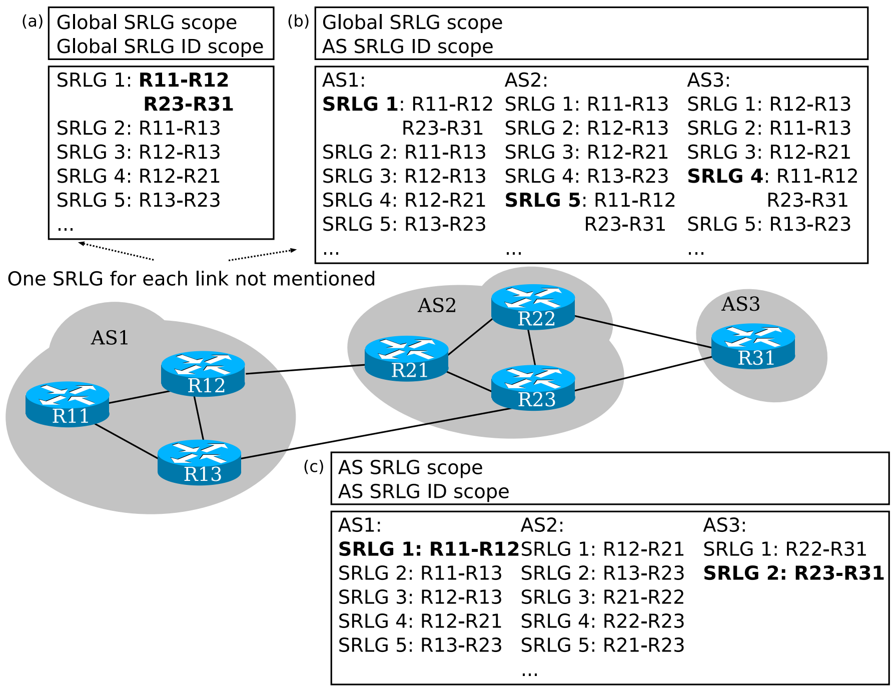
The motivation to support fast-reroute techniques as described in [PSA05] is twofold: first of all, it supports fast recovery, and, second, it provides SRLG protection, which is not the case for a disjoint end-to-end LSP. The problem to support SRLG protection, with the latter method, is described in this section.

There are different ways to provide end-to-end protection of inter-AS LSPs. A first possibility is to establish a secondary path that crosses different ASs than the protected LSP. An alternative is to establish an LSP that follows the same AS path to the destination as the protected LSP, i.e. it crosses the same ASs in the same order, but is link or node disjoint from the protected LSP. However, these two solutions do not enable to establish an LSP that is disjoint from the SRLGs of the protected LSP. That is, it is not possible to protect the primary inter-AS LSP against SRLGs failures with a single end-to-end link or node disjoint LSP. This is due to the fact that different ASs may possess links belonging to the same SRLG even if these ASs do not have the same convention to designate this SRLG. Links joining nodes that belong to different ASs may use the same resources. An example of such a situation is illustrated in figure 3.6 that is described later in this section.

We introduce the concepts of SRLG scope and SRLG ID scope to represent the set of nodes with a consistent view of the SRLG members and of their identification. All nodes in an SRLG scope see the same set of links belonging to that SRLG. The nodes in an SRLG scope will not be aware of links outside the SRLG scope that may share for instance physical resources with links in the SRLG scope. Hence links in the SRLG scope could fail at the same time as links outside the scope. These links belong to the same SRLG but the nodes in different scopes are not aware of it.

The SRLG ID scope represents the set of nodes with the same identifier for a given SRLG. Not all nodes in a particular SRLG scope must use the same SRLG ID to identify that particular SRLG. An SRLG scope can consist of different non-overlapping sections and each such section can use a different SRLG ID to refer to the SRLG. At the boundaries of these sections, there exists a one-to-one mapping of the corresponding SRLG IDs that identify the same SRLG.

Figure 3.6 illustrates the concepts of SRLG scopes and SRLG ID scopes. We have links $R11 - R12$ and $R23 - R31$ that belong to the same SRLG. They may fail at the same time. All the other links fail independently from the other links. They belong to SRLGs composed of a single link. In the top left frame (a), we consider a single SRLG scope that encompass $AS1$, $AS2$ and $AS3$. In this case, all the nodes inside these ASs are aware that links $R12-R21$ and $R23-R31$ belong to the same SRLG. In addition, the SRLG ID scope also covers $AS1$, $AS2$ and $AS3$. Thus, the SRLGs have the same identifiers for all the nodes in these ASs. SRLG composed of links $R12-R21$ and $R23-R31$ has identifier 1 for all the nodes in $AS1$, $AS2$ and $AS3$.



(a) Global SRLG scope
Global SRLG ID scope

SRLG 1: **R11-R12**
R23-R31
SRLG 2: R11-R13
SRLG 3: R12-R13
SRLG 4: R12-R21
SRLG 5: R13-R23
...

(b) Global SRLG scope
AS SRLG ID scope

AS1:	AS2:	AS3:
SRLG 1: R11-R12 R23-R31	SRLG 1: R11-R13	SRLG 1: R12-R13
SRLG 2: R11-R13	SRLG 2: R12-R13	SRLG 2: R11-R13
SRLG 3: R12-R13	SRLG 3: R12-R21	SRLG 3: R12-R21
SRLG 4: R12-R21	SRLG 4: R13-R23	**SRLG 4**: R11-R12 R23-R31
SRLG 5: R13-R23	**SRLG 5**: R11-R12 R23-R31	SRLG 5: R13-R23
...	...	...

(c) AS SRLG scope
AS SRLG ID scope

AS1:	AS2:	AS3:
SRLG 1: R11-R12	SRLG 1: R12-R21	SRLG 1: R22-R31
SRLG 2: R11-R13	SRLG 2: R13-R23	**SRLG 2: R23-R31**
SRLG 3: R12-R13	SRLG 3: R21-R22	
SRLG 4: R12-R21	SRLG 4: R22-R23	
SRLG 5: R13-R23	SRLG 5: R21-R23	
	...	

Figure 3.6: SRLG scope and SRLG scope ID

In frame (b) of figure 3.6, there is a single SRLG scope. This scope is divided into three SRLG ID scopes. Each SRLG ID scope covers an AS. Here, all the nodes know that links $R11-R12$ and $R23-R31$ may fail at the same time. However this SRLG is identified differently inside each AS. Thus, SRLG ID translation should take place at ASBRs. Using the same SRLG ID in different SRLG ID scopes does not mean that the SRLGs are linked to each other is some way.

In these two examples, all the SRLG link members are known by all nodes. Consequently, an end-to-end SRLG disjoint path may be computed by excluding the link belonging to the SRLGs crossed by the primary LSP. However, it is already difficult for Service Providers (SPs) to determine and maintain an up to date view of the SRLGs with scope limited to their AS [KYGS05]. It would be even harder for SPs to determine SRLGs with scope covering multiple ASs. This would require that SPs disclose their network topologies to other SPs despite the confidentiality requirement expressed in [RV05]. Moreover, the physical resources used by each link and their location would still have to be identified. Scopes that are not limited by ASs boundaries could be used for ASs that belong to the same SP. However, we do not assume that SRLG scopes covering multiple ASs will be defined in the majority of cases. Thus, in the remaining of this chapter, we consider SRLG scopes and SRLG ID scopes that are limited to single ASs.

When SRLG scopes do not contain all ASs crossed by the inter-AS LSP, an end-to-end recovery LSP crossing the same ASs may fail to provide SRLG protection as explained by the example that follows. Suppose that we have, in figure 3.6, a primary LSP going from $R11$ in $AS1$ to $R31$ in $AS3$ through $R12$, $R21$ and $R22$. It is not possible to protect this LSP against SRLG failures with a secondary LSP crossing for instance $R13$ and $R23$ when there are SRLGs with SRLG scopes corresponding to a single AS ($AS1$, $AS2$, or $AS3$), as in frame (c) of figure 3.6. Neither $AS1$ nor $AS3$ will be aware that links $R11 - R12$ and $R23 - R31$ may fail together. If an SRLG scope covers multiple ASs but not all the ASs crossed by the primary LSP, the same problem arises. The links in this SRLG scope may use the same physical resources as links outside the scope. This will not be known by the nodes in the different scopes. Consequently, these nodes may compute a secondary path that is not SRLG diverse from the primary path.

For the same reasons, it is not possible to ensure SRLG protection of a primary LSP with an end-to-end secondary LSP that does not share the same AS path if SRLG scopes do not cover the AS paths of both LSPs.

Because we consider SRLG scopes consisting of an AS, we focus on local protection, as defined in [PSA05], for SRLG protection. The solution proposed in this section enables the provision of link, node and SRLG protection of inter-AS LSPs.

In the next two sub-sections, we define one flag and one RSVP-TE subobject in order for our solution to support inter-AS SRLG protection.

SRLG Protection Desired Flag

[PSA05] does not specify how SRLG protection can be requested by the head-end LSR. One way to do this is to define an "SRLG protection desired" flag in the session attribute object. The head-end LSR sets this flag if it requires that the LSP be protected against SRLG failure.

In addition, we propose to define a new flag inside the RRO subobjects to indicates whether SRLG protection is provided for an LSP. In section 3.7.3, we explain that for node and SRLG protection, two detour LSPs or bypass tunnels are necessary. If these two detours or bypass tunnels are available to provide SRLG and node protection, then the “node protection” and “SRLG protection” flags are set in the corresponding RRO subobject. Similarly, the “bandwidth protection” flag of the RRO subobject [PSA05] is set when both detours or bypass tunnels provide the requested bandwidth. The setting of these flags informs the head-end LSR that the requested protection is ensured.

LSP-Merge Subobject

The LSP-Merge subobject is a new subobject in the Explicit Route Object (ERO). The procedures defined in [ABG$^+$01], section 4.3.4.1, to select the next hop are modified as follows: if after step 3 of the next hop selection process the node finds an LSP-Merge subobject in front of the ERO, i.e. the LSP-Merge subobject is the first subobject in the ERO after removing the subobjects belonging to the local abstract node, then the LSP has to merge with an LSP with the same Session object and LSP ID at the current node, if such an LSP exists. If no such LSP exists, then the detour LSP is rejected and a ResvErr with error code that is to be defined by the Internet Assigned Numbers Authority (IANA) is sent to the originating node.

The LSP with which the LSP containing the LSP-Merge subobject merges must be a protected LSP, i.e. it may not contain a DETOUR object. In addition the abstract node where the merging occurs must ensure that in case of a failure, the traffic can be switched from the LSP containing the LSP-Merge subobject to a recovery LSP that was established by the merging node to protect the working LSP. If these merging conditions cannot be met, the “SRLG protection” flag inside RRO subobjects, of section 3.7.2, is set to zero. This indicates to the head-end LSR that SRLG protection is not provided for the working LSP.

The need and usage of this subobject is illustrated later in this chapter.

3.7.3 Protection with Detour LSPs

In this section, we show how to protect an inter-AS LSP against the failure of the resources at the boundaries of the ASs with detour LSPs. First, we consider the protection against inter-AS link failures. Then, we look at the protection of ASBRs. Finally, we present techniques for inter-AS SRLG protection by means of detour LSPs.

We describe, for each type of resources to protect, the actions to be performed at the different LSRs. We provide the actions required at the egress ASBR, at the ingress ASBR, at the secondary egress ASBR and at the secondary ingress ASBR.

Link Protection with Detour LSPs

The primary egress ASBR establishes a detour LSP to protect its downstream inter-AS link. The destination of the detour LSP is the same as the destination of the protected LSP. The detour LSP may merge with the protected LSP at any downstream node or with other detour LSPs of the same protected LSP, established by nodes downstream of the link to be protected.

The egress ASBR has to determine a secondary egress ASBR and then it can perform a path calculation towards this ASBR. The primary egress ASBR can select any other ASBR as secondary egress ASBR. However, it is recommended to select an ASBR that is connected to the downstream AS of the protected LSP (i.e. the AS where the primary ingress ASBR is located). More precisely, we advise to use one of the inter-AS links of this secondary egress ASBR that is connected to the downstream AS of the protected LSP. Bonaventure et al. [BFF05] propose a solution to discover such inter-AS links. In case this condition is not met, it could be for instance possible that the downstream AS of the detour LSP chooses a path that goes through the AS where the detour LSP was originated causing loops. This is illustrated in figure 3.7. Suppose the protected LSP crosses the domains $AS1$, $AS2$, $AS3$ and $AS4$ in that order. The detour LSP protecting the link $R22 - R31$ between $AS2$ and $AS3$ does not take an alternative link between $AS2$ and $AS3$ but it takes link $R24 - R61$ toward $AS6$, then $AS6$ could take $AS5$ as next AS and then at the end the detour LSP arrives at $AS1$ where it merges with the protected LSP. It is clear that such detour does not protect the link that it is supposed to protect. Note that it is only a recommendation and not a must to take the same downstream AS because there are ways to solve this problem by excluding ASs [LFC05] but this could be a rather complex solution[8].

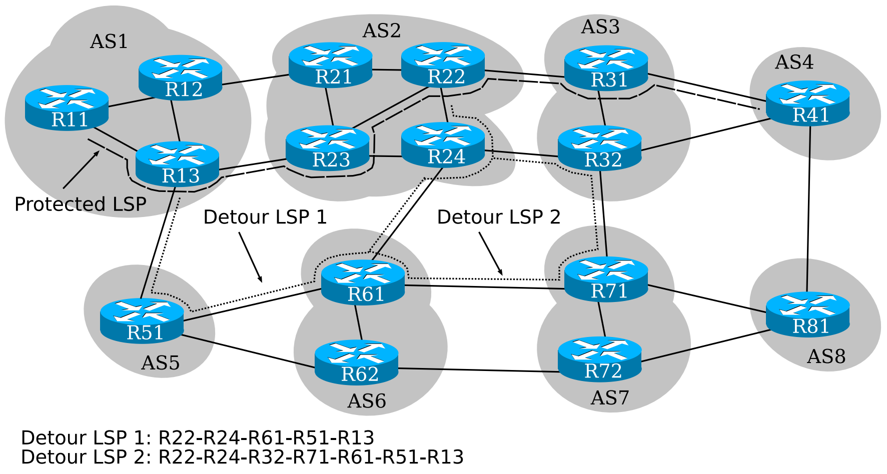


Figure 3.7: Guidelines for correct detour LSP merging

In addition, we recommend that the detour LSP merges in the AS where the downstream ingress ASBR is located (the merging node could be the ingress ASBR itself) if the destination of the protected LSP is not in the downstream AS. For example, in figure 3.7, the detour LSP protecting the link $R22-R31$ of the working LSP between $AS2$ and $AS3$ should merge with the working LSP in $AS3$. If this does not happen, $AS3$ could select $AS7$ as next AS for the detour LSP and from then on $A7$ could select $AS6$, which further goes to $AS5$ and $AS1$ where the detour LSP merges with the working LSP upstream from the failure to protect. This recommendation also improves the scalability of the solution since merging LSPs reduces the number of states to be maintained, the bandwidth to be reserved, and so on.

[8] If AS Numbers (ASNs) are recorded inside the RRO, as proposed in 3.6.3 for RRO aggregation, the solution would not be so complex.

To enforce the recommendations of the previous paragraphs, the ERO for the detour LSP starting at the egress ASBR should contain several path's segments. It should first contain a strict or a loose path towards the secondary egress ASBR followed by a segment of the RRO of the protected LSP. The latter segment begins at the last hop in the downstream AS (the egress ASBR in the downstream AS) of the protected LSP and contains all hops thereafter up until the destination. This ensures merging of the detour with the protected LSP at the latest at the egress ASBR in the downstream AS. That is, the ERO of the detour LSP at least contains:

1. A strict or loose path toward the secondary egress ASBR.
2. The path of the working LSP starting at the last hop inside the downstream AS and ending at the destination of the working LSP.

In the example network of figure 3.7, we have a protected LSP with path $R11 - R13 - R23 - R22 - R31 - R41$. Suppose that the selected egress ASBR is $R24$ and the calculated path towards $R24$ is $R22 - R24$ ($R22$ is originator of detour LSP). The ERO of the detour LSP protecting link $R22 - R31$ should therefore be composed of router $R24$ (strict hop).$R24$ is followed by $R31$ (with loose flag set) and the next routers after $R31$ on the path of the working LSP in the downstream AS of $AS3$, that is $R41$. All these nodes are obtained from the information recorded in the RRO. We note that the path between $R24$ and $R31$ has to be calculated by $R24$ and $R32$. Each of these routers computes a portion of the path.

There are two possible methods to determine the secondary egress ASBR at the primary egress ASBR. First, the egress ASBR can be manually configured with other ASBRs that peer to the same AS. Second, it can lookup in its BGP table to find an other entry such that the AS-path has the same AS next hop as the currently selected entry. The first option is feasible because the number of links between 2 ASs is usually limited to only a small number of links. The second option may not provide an alternate egress ASBR because of the lack of diversity in the BGP routes [PUB04, UT06].

It could be possible that the primary egress ASBR is the same router as the secondary egress ASBR and that the primary ingress ASBR is the same router as the secondary ingress ASBR. In this particular case when there are multiple links between the ASBRs, the detour LSP must simply use an inter-AS link that is not the one used by the working LSP, and no path computation has to be done at the egress ASBR.

The use of the LSP-Merge subobject, defined in section 3.7.2, is optional to provide link protection. This is an ERO subobject that forces the merging at the next node in the ERO and it makes sure that this merging node can switch traffic coming from the merging detour LSP to the originating detour LSP.

No specific procedure is required at the primary ingress ASBR. We note that the detour LSP may merge with the protected LSP at this node.

The secondary egress ASBR completes the path in the ERO by selecting a secondary ingress ASBR in the downstream AS. If there is no ERO present, then the tunnel end point address in the Session object has to be used to route the Path message.

The secondary ingress ASBR completes the ERO with a path towards the next subobject in the ERO. The LSP should merge with the working LSP at the node that processes the LSP-Merge subobject (if that subobject is present), if it was not yet merged at this point. If no ERO is present inside the Path message of the detour LSP, the path is computed based on the tunnel end point address.

Node Protection with Detour LSPs

The procedures and recommendations are the same for the protection of an ingress ASBR failure as for link protection, with the exception that the egress ASBR has to include an XRO object or an EXRS subobject [LFC05] with the ingress ASBR to exclude.

For the protection of the egress ASBR, the procedures described earlier for the egress ASBR now apply to the router on the path of the protected LSP that precedes the egress ASBR. This router is either directly connected to the egress ASBR or it is the upstream ingress ASBR if the LSP is tunneled inside an intra-AS LSP.

The method to determine a secondary egress ASBR is the same as for link protection: either manual configuration or by using BGP routing information, if it is available. Note that the first solution requires more configuration as for link protection in case this router peers with more than one ASBR.

SRLG Protection with Detour LSPs

Similar procedures as for link protection apply for SRLG protection. In addition, the secondary egress ASBR must absolutely be an ASBR that peers with the downstream AS of the protected LSP. And, the detour LSP must merge in that AS. The former condition is necessary because only the two peering ASs know the SRLGs of the inter-AS link and the latter condition implies that the LSP-Merge subobject must be used. This subobject is inserted inside the ERO to indicate the node where merging needs to be done (see section 3.7.2). The remaining of this section describes in more details the procedures to be used at the nodes involved in the establishment of such detour LSP.

The egress ASBR has to include an XRO object or an EXRS subobject to exclude the SRLGs of the inter-AS link. The XRO or the EXRS must include a list of SRLGs (defined for the AS containing the PLR) corresponding to the inter-AS link as well as a reference to this link. This reference is used to map the SRLG IDs defined in one AS into the respective SRLG IDs allocated in another AS. If the egress ASBR can compute a strict path to reach the secondary egress ASBR, then the list of SRLGs does not need to be included. Only the reference to the link for which the detour LSP has to be SRLG disjoint is then required. The secondary ingress ASBR has to use the information in the XRO or EXRS to further compute a path for the detour LSP.

To ensure merging inside the downstream AS, the LSP-Merge subobject (see section 3.7.2) has to be included in the ERO by the egress ASBR. The LSR where the detour LSP is merged with the protected LSP has to ensure that it can perform a switch-over from the incoming detour LSP containing the LSP-Merge subobject to its originating detour LSP in case the next link has an SRLG in common with the inter-AS link. This is because in this case, both links can fail at the same time such that both detour LSPs will be activated at the same time. In other words, the PLR has to send the traffic from the main LSP or the incoming detour LSP on the departing detour LSP protecting the failure of the downstream resources, when protection is in use.

No extra procedures are required at the primary ingress ASBR.

The secondary egress ASBR selects a next hop for the detour LSP. The XRO or EXRS contains a reference to the link for which the detour LSP has to be SRLG disjoint. No list of SRLGs should be included because the SRLG IDs are local to an AS, which means that if a list of SRLG IDs would be sent to the next hop, then this node would not understand the IDs. Therefore only the reference to the inter-AS link is useful. This link is referenced by means of its IP address, see [LFC05]. The secondary egress ASBR thus removes the list of SRLGs related to the inter-AS link, if such a list of SRLGs is present in the XRO or EXRS.

If the secondary ingress ASBR cannot compute a full path towards the node immediately preceding the LSP-merge subobject, then the secondary ingress ASBR adds the list of SRLGs of the inter-AS link to the received XRO object or EXRS subobject, respectively. These SRLGs are known by the nodes inside this AS. This is required because the LSP can cross nodes inside the AS which do not know the SRLGs of the inter-AS link, but only the SRLGs of intra-area links, hence just a reference to a link whose SRLGs have to be excluded is not sufficient. An alternative would be to distribute inter-AS links and their SRLGs inside the IGP.

To allow the egress ASBR and the secondary ingress ASBR to calculate a path, the SRLGs of the inter-AS links towards the same downstream AS (upstream AS, respectively) as the working LSP have to be known. This could be achieved through manual configuration of the SRLGs of other inter-AS links to the same downstream/upstream AS at each ASBR. For instance, in figure 3.7, at $R24$ and $R32$, the SRLGs of $R22 - R31$ can be configured such that they are known for the path calculation, and at $R22$ and $R31$, the SRLGs of $R24 - R32$ can be configured. An other option is to flood this information via BGP extensions still to be defined or to distribute these links and their SRLGs inside the IGP.

We do not assume that nodes other than ASBRs having a link to the same downstream/upstream AS know the SRLGs of these inter-AS links. If this would be the case, then the procedures above could be simplified, e.g., the egress ASBR would not have to include a list of SRLGs anymore when only it only computes a partial toward the secondary egress ASBR.

In addition, the secondary egress ASBR has to know the SRLGs of the inter-AS link used by the protected LSP. This enables it to select a link in case there are multiple links towards the downstream AS, and to check if the selected link is indeed SRLG disjoint from the inter-AS link used by the protected LSP.

SRLG and Node Protection with Detour LSPs

In this section we first consider the protection of the egress ASBR and of the SRLGs of the link preceding this ASBR. The SRLG protection of the other intra-domain links and their downstream node is solved by [PSA05]. Then, we present the procedures for SRLG protection of the inter-AS link and node protection of the downstream ingress ASBR.

If the inter-AS LSP to be protected is tunneled inside an intra-AS LSP, the intra-AS LSP has to be protected against the failure of the resources along its path. Thus, if SRLG and node protection is required, the intra-AS LSP has to be protected against the failures of each SRLG and each node that it crosses with the exception of its head-end and tail-end LSRs. Consequently, the intra-AS LSP will be protected against the SRLG failures of its last link. Therefore, in case of tunneling, the procedures exposed in this section will not be used.

Protection of the egress ASBR and SRLG protection of the link preceding the egress ASBR is best solved by using two detour LSPs at the node on the path of the working LSP preceding the egress ASBR: a detour to protect against the SRLGs of the intra-AS link and a second detour LSP that is established using the procedures for node protection described earlier. The detour protecting against the SRLGs has to merge in the same AS, i.e. it has to merge with the protected LSP at the egress ASBR. This is because other ASs do not know this intra-AS link, nor its SRLGs. To ensure that merging occurs at the egress ASBR, the RRO of the protected LSP should be fully included in the ERO of the detour LSP together with an LSP-Merge subobject that is inserted after the subobjects representing the egress ASBR. The ERO should be further prepended by a path, which is SRLG disjoint with the downstream link of the PLR on the protected LSP (i.e. the intra-AS link), computed towards the egress ASBR. This could only be a partial path towards the egress ASBR in which case an XRO object or an EXRS subobject, containing the SRLGs to avoid, has to be added. Moreover, it has to be ensured that these 2 detour LSPs do not merge. Otherwise, SRLG protection could not be guaranteed. This means that at least one of the detour LSP should be a sender-template specific detour LSP.

The egress ASBR must ensure that it can do a switch-over from the incoming detour LSP protecting against a failure of the preceding link to its originating detour LSP. This is because the preceding link and the inter-AS link can belong to the same SRLG, hence they can fail at the same time. For this reason, the LSP-Merge subobject must be used.

If protection of the ingress ASBR is requested, in addition to SRLG protection of its upstream link, the egress ASBR also has to put the ingress ASBR in the XRO or EXRS like it was done for node protection. The use of 2 detour LSPs (one for SRLG protection and the other for node protection) is also recommended when the ingress ASBR is to be protected. If only 1 detour LSP is used, we can have the following problem. If the protected LSP only crosses 1 hop in the downstream AS (i.e. ingress ASBR and egress ASBR in the downstream AS are the same router), the detour LSP setup would not be able to provide SRLG protection. This is because the detour LSP crosses 3 ASs in this case: the AS where it originates, the single-hop AS (the hop to be protected), and the AS where it merges again. However, the latter AS is not anymore aware of the SRLGs of the link to be protected because that link is between the first two ASs. For example, suppose that the protected LSP in figure 3.7 traverses nodes $R51 - R61 - R71 - R81$. Node $R61$ together with the SRLGs of $R51 - R61$ have to be protected. A single detour LSP protecting the SRLGs and $R61$ would traverse $R51 - R62 - R72$. However, $R72$ in $AS7$ cannot further expand the ERO of the detour LSP because it does not know the SRLGs or the SRLG IDs of $R51 - R61$ between $AS5$ and $AS6$ (assuming local SRLGs). Therefore, 2 detour LSPs must be used: a detour LSP traversing $R51 - R62 - R61$ for SRLG protection, and a detour LSP traversing for instance $R51 - R62 - R72 - R71$ to protect $R61$.

In case of node and SRLG protection or in case of SRLG protection only, it is required to use sender-template specific detour LSPs to avoid that detour LSPs merge with each other.

3.7.4 Protection with Bypass Tunnels

The problem of protection by means of bypass tunnels can be split into two components:

1. The bypass tunnel has to be signaled over a path that is disjoint with the network resources that it protects.

2. After the bypass tunnels are established, an appropriate bypass tunnel has to be selected for each particular working LSP such that the protection requirements for that LSP are met.

The first part is very similar to the establishment of detour LSPs: an XRO object or an EXRS subobject can be used to signal the bypass tunnel such that it is disjoint from the network resources used by the protected LSP. The same recommendations as for detour LSPs apply. It is recommended that the downstream AS of the bypass tunnel and of the working LSP are the same AS. Additionally, as two detour LSPs are required for SRLG protection of the upstream link of an egress ASBR and the egress ASBR itself, two bypass tunnels are also required to protect these resources. Note that the LSP-Merge subobject is not used for bypass tunnels. Bypass tunnels do not merge with the protected LSP at the far-end of the bypass tunnel. They are terminated at that node.

The difficulty in providing protection with bypass tunnels lies in the selection of appropriate bypasses for the protection of given resources. To select a bypass tunnel, the PLR has to take a bypass tunnel that it originates and that fulfills the following requirements:

1. The bypass tunnel must fulfill the appropriate constraints (bandwidth, link affinities, ...).

2. The bypass tunnel must be disjoint with the link/node/SRLGs to be protected.

3. The destination of the bypass tunnel must be the next-hop node (resp. next-next-hop node) of the protected LSP, or a node further downstream on the path of the protected LSP, in case of link protection (resp. node protection).

The first two requirements can be achieved because the required information is locally available at the PLR. The PLR has established the candidate bypass tunnels, hence it knows the bandwidth and the resources protected by the bypass tunnel. Complying with the third requirement is more difficult. Generally, the PLR must check if the destination of the bypass tunnel belongs to one of the nodes listed in the RRO of the Resv message of the protected LSP. Usually the RRO contains interface addresses. However, the destination of a bypass tunnel may be a different interface address or the node-id of the router. This means that the PLR has to map the addresses listed in the RRO of the protected LSP to the destination address of the bypass tunnel. In an intra-area environment this is possible since this information is available in the IGP topology, but in the inter-AS case, this information is not available locally at the PLR. There are several methods to solve this problem:

Solution A : use the solution described in [VAS06] where the node-id of the routers are placed in the RRO of the Resv message of the protected LSP and the node-id is also put in the RRO of the Resv message of the bypass tunnel if the destination used to establish the bypass was not the node-id. In this way, the PLR simply has to compare the node-ids in the RRO of the protected LSP with the destination of the bypass tunnel or with the node-id of the destination in the RRO of the bypass tunnel.

Solution B : use the interface address that would be recorded in the RRO of the protected LSP as the destination of the bypass tunnel. For instance, when the link between two ASBRs, $ASBR1$ and $ASBR2$ is to be protected, the destination address would be the address of the interface on $ASBR2$ towards $ASBR1$. If this link is unnumbered, the destination address used is the node-id that is also mentioned in the RRO of the protected LSP. This is sufficient to identify the common node on the protected LSP and the bypass tunnel. When node protection is to be provided and the destination of the bypass tunnel is the next-hop of the protected node (next-next hop from the PLR point of view), the destination of the bypass tunnel should be the address of the interface on the next-next-hop router that goes towards the node being protected. Consequently, multiple bypass tunnels must be used in case of parallel links. We note that although the interface is used as destination, the bypass tunnel enters the node via another link and a failure of the interface used as destination of the bypass tunnel must not lead to the failure of the bypass tunnel itself (this is in particular important for link protection).

Until now, we supposed that the bypass tunnels were manually configured, with the destination being part of the configuration. But, bypass tunnels can also be signaled automatically when the first protected LSP is established. Therefore, we have to determine the destination of these dynamically established bypass tunnels. In case of solution B, the information about the interface addresses in the RRO of the protected LSP can be used as a destination address. In case the node-id is put in the RRO, then this node-id can be used.

3.8 End-to-end Disjoint Inter-AS LSPs

In this section we describe the establishment of end-to-end links or nodes disjoint LSPs for link/node protection or load balancing purposes. We present a technique to compute two disjoint paths. However, it can be applied for any number of disjoint paths computation. We do not assume the existence of a per-domain entity (or multiple entities per-domain) responsible for the path computation on behalf of all the ingress ASBRs of the domain along the disjoint LSPs. Thus, we do not consider a simultaneous computation of the disjoint paths and we do not require that the disjoint paths follow the same AS path. We note that the consecutive computation of the disjoint paths is subject to the trapping problem[9]. However, [Gro04] explains, in page 209, that this problem is unlikely to occurs in transport networks.

We distinguish two cases:

1. The disjoint LSPs do not have the same AS path.

2. The disjoint LSPs have the same AS path. That is they cross the same ASs in the same order.

The first case is simpler to solve than the second one. If the second LSP crosses an AS that is not crossed by the first LSP, the path computation for the second LSP does not need to take into account the path of first LSP, except for selecting the ingress ASBR inside the downstream AS.

However, if the second LSP crosses an AS that is used by the first LSP, the links and nodes along the first LSP need to be excluded from the computation of the second LSP. However, the nodes that participate in the computation of the path for the second LSP are not crossed by the first LSP, due to the disjointness constraint. Thus, these nodes do not know the path of the first LSP inside the AS. We assume that they learn a summary of the first path from the XRO. The XRO is formed based on the RRO of the first LSP. Since we introduced RRO aggregation and intra-AS tunneling to enforce the confidentiality requirement of SPs, the RRO

[9]The trapping problem concerns the fact that the path chosen for an LSP may not allow to fi nd a disjoint path even if two disjoint paths exist

contains a summary of the LSPs path. The intra-AS paths do not appear in the RRO. As a consequence, the nodes on the path of the second LSP do not know the links and nodes to be avoided by the second LSP.

The ingress ASBRs inside each AS are stored inside the RRO with both RRO aggregation and intra-AS tunneling. Thus, if the XRO contains the information gathered by the RRO, each node along the second LSP learns the ingress ASBR inside each AS. Inside each AS, the ingress ASBR of the first LSP knows the sequence of links and nodes along the path of this LSP. This information is gathered by the RRO inside the Resv message and stored inside the state maintained for the LSP. We propose to retrieve this information when a node along the second LSPs has to compute a segment of the path. We suggest to extend the communication protocol proposal of [VLA$^+$06] in order for this node to retrieve the path of the first LSP based on the identifier of the LSP. Once this path is known it can compute a disjoint portion of the path inside the local domain and select a different downstream ingress ASBR (depending on the topology information available at the node).

3.9 Related Work

When the work in this chapter was published in [PB02], [PB03] and [CP04], there were very few papers discussing solutions to allow the establishment of LSPs across AS boundaries. We briefly describe this work in the following paragraphs.

Okumus et al. [OHMC01] propose a solution based on the utilization of a specialized Bandwidth Broker agent relying on the SIBBS inter-domain signaling protocol. Our solution based on RSVP-TE has two main advantages over the utilization of a special inter-AS signaling protocol. First, RSVP-TE is already implemented and deployed, which is not the case of SIBBS. Second, our extensions can be added to existing RSVP-TE implementations with a limited amount of effort.

Another solution is the utilization of the BGP extension defined in [RR01] to distribute MPLS labels and thus establish inter-AS LSPs. Compared with our solution, a drawback of this BGP approach is that with BGP, the inter-AS LSPs are established without being able to specify bandwidth or fast restoration constraints.

Blanchet et al. [BPSA01] propose two BGP extensions to allow the establishment of optical inter-domain paths. The first extension allows to distribute reachability information by defining a new BGP multi-protocol extension and using extended communities to encode lightpath information. The second extension proposes to use BGP to setup inter-domain lightpaths. This setup is based on the utilization of a BGP update messages containing special extended communities. This second extension has several drawbacks compared to our solution. First, [BPSA01] only addresses the signaling of the lightpath between domains, it does not discuss how an inter-domain path should be established inside each transit domain while our solution works both inside and outside domains. Second, the BGP extensions described in [BPSA01] do not allow to specify fast restoration or QoS requirements for the path being established.

The work on the protection of MPLS LSPs was focussed on the protection of intra-area and intra-AS LSPs at the time [PB02], [PB03] and [CP04] were published. Among this work there is [PSA05] that provides extensions to RSVP-TE for the local protection and fast-reroute of intra-AS LSPs. Later, interest focussed on inter-area protection. [LVB05] expresses requirements for inter-area TE with MPLS. [RD04] provides a solution to protect an LSP crossing multiple areas with another LSP that is end-to-end disjoint from the protected LSP. Then, requirements were expressed in [RV05] for inter-AS TE with MPLS. Tools for the fulfillment of these requirements were then developed. [AV06a] defines extensions to RSVP-TE for the establishment and the maintenance of LSPs that cross multiple domains. In addition, [VIZ06] proposes ways to trigger the reoptimization of intra and inter-AS LSPs.

Besides the work on the protection of traffic carried in MPLS LSPs, there is also interest in the research community on minimizing traffic disruptions in pure IP networks. Inside ASs, modifications to the Interior Gateway Protocol (IGP) are proposed to improve restoration in case of a maintenance or failures. [FB05] proposes a solution to avoid transient loops during the reconvergence of the IGP in order to avoid packet loss. [KHC$^+$06] makes use of the capability of the OSPF and IS-IS to handle multiple topologies. The authors propose an algorithm to compute backup routing tables. When a failure occurs, one of these tables is selected and traffic is forwarded according to this routing table.

We also note the interest of the community on BGP convergence time reduction and on the protection of BGP peering links. The solution described in [CDZK05] aims at reducing the exploration of the interdomain routes with BGP upon the occurrence of a failure. For this purpose, “path dependency” information is advertised in the BGP routes. This information is used to determine the routes that should not be considered as candidates for BGP route selection when a failure occurs. This results in a faster convergence of BGP. In [BFF05], peering links are protected by means of IP tunnels. An ASBR selects the destination of its protection tunnel based on information that is obtained from a BGP auto-discovery mechanism proposed by the authors.

3.10 Conclusion

Although MPLS and GMPLS are currently used only inside ASs, applications such as inter-AS VPNs, inter-AS fast-restoration and traffic engineering force network operators to also consider those technologies across AS boundaries.

In this chapter, we have shown how it is possible with RSVP-TE to establish intra-AS LSPs that avoid hot-spots in the network. The LSPs may follow a specific path that is totally or partially pre-configured or computed at the LSPs head-end. Moreover, these LSPs can be protected in order to recover quickly from failures. Then, we have discussed the requirements for the establishment of inter-AS LSPs.

We have proposed extensions to RSVP-TE to be able to establish TE LSPs toward a prefix and an AS destination in addition to an IP destination. These extensions enable to perform prefix-based and AS-based TE without requiring the knowledge of a specific IP node address inside the prefix and AS destinations. We have defined the notions of SRLG scope and SRLG ID scope. We have shown that SRLG protection of inter-AS LSPs with SRLG and SRLG ID scopes limited to a single AS can only be provided with local protection. We have then showed that by introducing a limited number of protocol extensions, it is possible to establish inter-AS LSPs with local protection while still preserving the confidentiality requirement of network operators. Our protocol extensions support Bypass tunnels, Detour LSPs and also allow to establish disjoint inter-AS LSPs for load balancing or end-to-end restoration.

Chapter 4

Computation Techniques for Interdomain Constrained LSPs

As we have already mentioned in chapter 2, BGP only provides reachability information for the destinations. More precisely, it only provides the addresses of Next Hops (NHs), the nodes at the border of the domain, that are able to forward the packets to a given destination. The QoS properties of the paths, such as the delay and bandwidth, behind these NHs are not provided. This results in several limitations for the computation and establishment of constrained interdomain LSPs.

Inside an AS composed of a single IGP area, all routers learn the complete topology of the AS by means of ISIS/OSPF. Thus, each router is able to compute the complete path from head-end to tail-end node for an LSP contained in the AS. However, the topology of an AS is hidden to routers outside the AS, notably for confidentiality purposes [SMWA04]. As a consequence, a single node is not able to compute an end-to-end path composed of strict, non abstract node for an LSP crossing multiple ASs. Therefore, the computation of such a path has to be distributed among multiple nodes, where each node computes a segment of the path based on its knowledge of the local AS topology and the interdomain reachability information provided by BGP.

In this chapter, we present the current state of development of two distributed techniques for the computation of paths respecting QoS requirements together with a centralized technique that we will use in subsequent chapters as a reference point to compare with the other techniques. In appendix A, we present our implementation of these techniques in a simulator. We describe in the appendix the choices that we made concerning points that were not described in the IETF documents at the time of the implementation or are still not described today.

In the next section, we introduce the notion of Path Computation Element and the capabilities of such an element. This element is used in several path computation techniques that are described in this chapter.

4.1 Path Computation Element

The Path Computation Element (PCE) [FVA06] may be any Label Switching Router (LSR), an Area Border Router (ABR), an AS Border Router (ASBR) or a dedicated server.

The role of the Path Computation Element (PCE) is to participate in the computation of constrained paths. Each PCE is assigned a domain and can compute constrained paths segments within its domain [1]. We call these segments, "local" path segments. The PCE can also compute a path based on local path segments and path segments received from other PCEs.

When the head-end and the tail-end of the LSP do not belong to the same domain or, if the LSP has to cross different domains, computation of the path for the LSP is distributed. Multiple PCEs may contribute to the computation of the end-to-end path.

The domain of a PCE may span a single or multiple areas, an AS or multiple ASs. In this thesis, we consider that the domain of a PCE is an AS. There is at least one PCE inside each AS. For clarity reasons, we describe the following path computation techniques based on the existence of a single PCE inside each AS. However, they are also applicable (1) when there are multiple PCEs inside ASs with domain covering an AS as well as (2) when the ASs are divided in more than one IGP area, there is one PCE for each area and the domain of each PCE covers an area.

The PCE computes path segments respecting given QoS and diversity constraints based on a Traffic Engineering Database (TED). The content of the TED for inter-domain Traffic Engineering (TE) is still under discussion at the IETF [FVA05]. It depends on the domain of the PCE. The TED contains at least the topology of the domain and the TE attributes of the links belonging to the domain. In addition, it may contain the TE attributes of the links at the border of the domain, for example the inter-AS links. This information is distributed by the Traffic Engineering (TE) extensions to the Interior Gateway Protocols (IGP) [KKY03], [SL04]. Moreover, the TED must also contain reachability information for destinations outside the domain of the PCE. This information is distributed by the Border Gateway Protocol (BGP) for destinations outside the AS.

Nodes requesting a path computation from a PCE are called Path Computation Clients (PCC). Thus, a PCE asking for a computation from another PCE acts as a PCC. The communication protocol between PCCs and PCEs, meeting the requirements expressed in [AL06], is defined in [VLA$^+$06]. The message sent by a PCC to a PCE, requesting a path computation is noted PCReq. The response of the PCE to the PCC is a Path computation Reply, PCRep.

An additional mechanism, called PCE discovery, is required in order for the PCCs to learn the list of PCEs that are available in their domain and in neighboring

[1]These segments start at the entrance of the domain or at the head-end of the LSPs. They end at the entrance of the downstream domain, at the egress node of the current domain or at the tail-end of the LSPs.

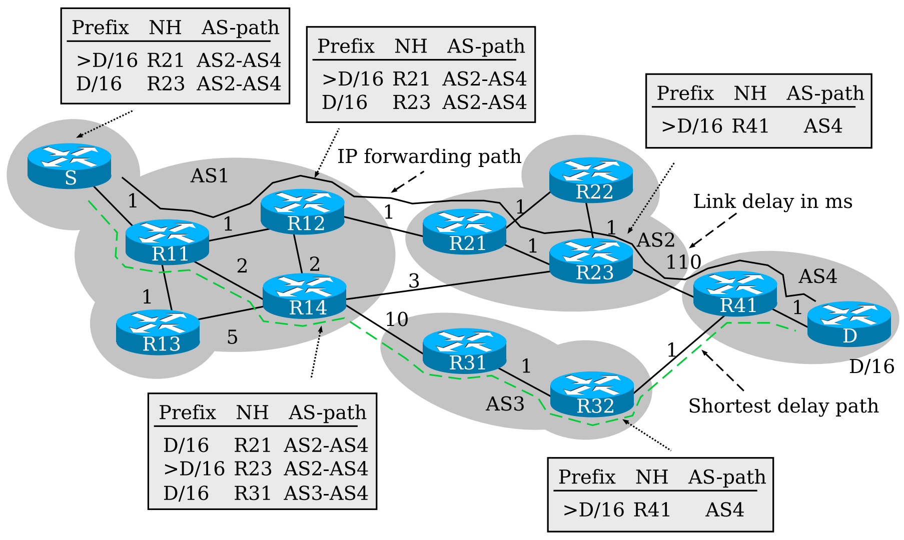
domains[2]. The requirements for such protocol are expressed in [LMO+06].

4.2 Computation techniques

4.2.1 Standard IP Forwarding

The simplest technique to establish an interdomain MPLS LSP is to follow the same path as the normal IP packets. This path is determined by the IGP inside an AS and by BGP for destinations outside the AS. This would be the path chosen by the Label Distribution Protocol (LDP) if LDP were used between ASs. The drawback of the IP forwarding path toward a destination is that it may not respect given QoS constraints.

Figure 4.1 shows the IP forwarding path from node S to node D. This path crosses nodes $S - R11 - R12 - R21 - R23 - R41 - D$. More precisely, S determines that $R21$ is the BGP NH to reach nodes belonging to prefix D/16, based on its forwarding table. From its forwarding table, S also knows that $R11$ is on the shortest path to $R21$. Thus, it sends IP packets to $R11$. The same process takes place at each router on the forwarding path, until the destination is reached. We note that the end-to-end delay along the IP forwarding path is equal to $115ms$. Thus, it could not support a flow requiring a delay below $100ms$. There is a shorter path, going through $S - R11 - R14 - R31 - R32 - R41 - D$, respecting this end-to-end delay constraint. However, this path is not available at node S.



Prefix	NH	AS-path
>D/16	R21	AS2-AS4
D/16	R23	AS2-AS4

Prefix	NH	AS-path
>D/16	R21	AS2-AS4
D/16	R23	AS2-AS4

Prefix	NH	AS-path
>D/16	R41	AS4

Prefix	NH	AS-path
D/16	R21	AS2-AS4
>D/16	R23	AS2-AS4
D/16	R31	AS3-AS4

Prefix	NH	AS-path
>D/16	R41	AS4

Figure 4.1: IP forwarding

[2]Discovery of PCEs in neighboring domains is required for path computation by means of cooperative PCEs. This technique is presented in section 4.2.4.

4.2.2 Centralized Path Computation

In this technique, the computation is performed by a single entity, that we name "global PCE". The domain of the global PCE covers all the ASs crossed by the LSPs. We assume that the global PCE learns the complete topology by receiving the ISIS/OSPF link state packets of each AS. The existence of tools such as [ZK05], [Bon04] and [SG04] proves this to be feasible. The global PCE performs a path computation for each LSP. We note that such a computation does not rely on BGP. It is not constrained by BGP peering relationships and route filtering. The algorithm used by the global PCE for path computation is the CSPF algorithm, in this thesis. However, other algorithms could be envisaged depending on the TE objectives to achieve and the algorithm complexity that is tolerated. This CSPF computation by a global PCE provides an indication of the path quality that can be achieved with a centralized computation.

Such a centralized solution could be envisaged when MPLS LSPs are entirely contained inside ASs that belong to the same company. However, it is not realistic for MPLS LSPs that cross ASs from different companies as this requires the ASs to cooperate and reveal their internal topology. Moreover, this solution is not scalable in the number of nodes and links of the ASs considered by the centralized computation. We use it as a benchmark and compare it with more easily deployable techniques.

In figure 4.2, we assume that the domain of the PCE covers $AS1$, $AS2$, $AS3$, $AS4$ as well as their interconnections. When an LSP is required between the head-end S and the tail-end D, S sends a PCReq message to the PCE. The constraints required for the LSP are specified in this message. Upon reception of the PCReq, the PCE computes a path that respects the constraints. Then, it sends the path back to the PCC, S in a PCRep message. Here, we see that the path returned by the PCE is the path with shortest delay because the PCE uses the CSPF algorithm.

4.2.3 ERO Expansion

Because the use of a global PCE is not applicable in the general interdomain framework, other techniques are required. In this section, we consider a distributed path computation method. This technique relies on the simultaneous computation and establishment of interdomain MPLS LSPs. It makes use of RSVP-TE to establish interdomain MPLS LSPs and of its ability to crankback. That is the capability (1) to stop the establishment of an LSP at a node when it cannot compute a path toward the destination that respects the constraints of the LSP and (2) to establish the LSP along a different path.

Inside RSVP-TE, it is possible to indicate the path or a portion of the path to be followed by the LSP inside an object called the Explicit Route Object (ERO) (see section 3.2). The ERO expansion technique, described in [VAZ06], relies on this object. It consists in completing at the ingress router of a domain, the ingress AS Border Router, the path computation up to the BGP Next-Hop (NH),

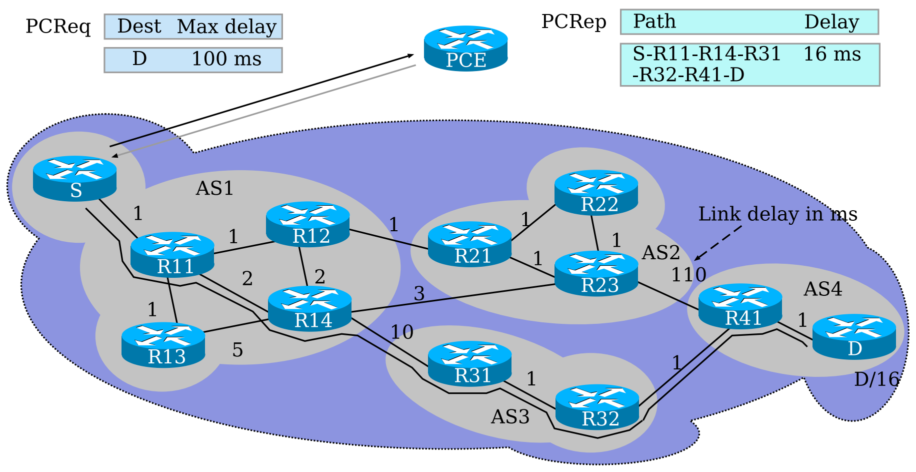


Figure 4.2: Centralized computation with a global PCE

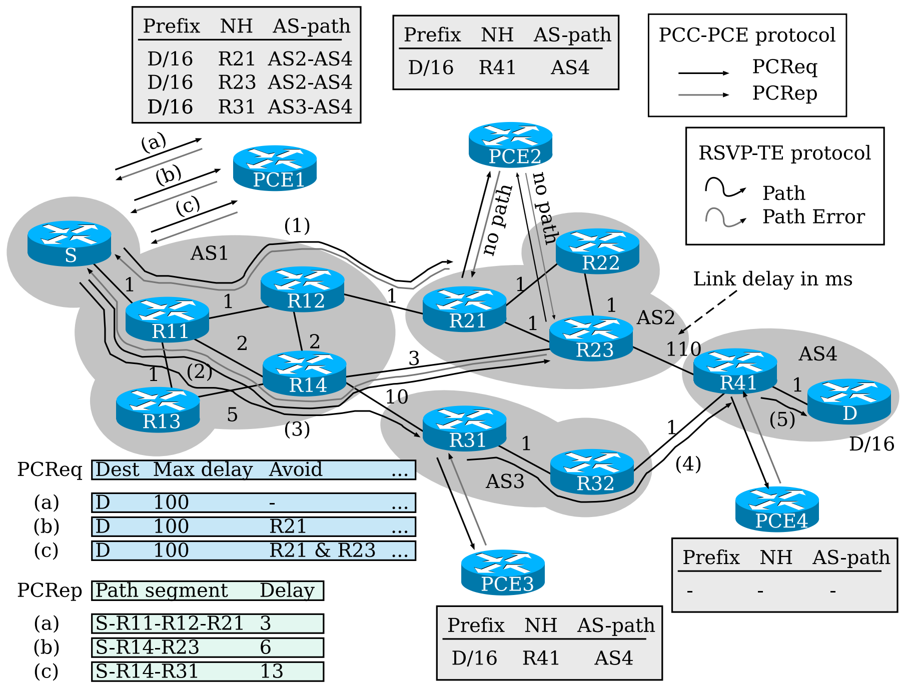
i.e. last reachable hop toward the destination. This node is either the first hop inside the downstream domain or the last hop inside the current domain. The computed path segment is then stored inside the ERO of the RSVP-TE Path message. This message is forwarded along the path specified inside the ERO and requests the establishment of the LSP along the path.

We propose that a dedicated PCE be responsible for the computation of the paths on behalf of the ingress routers in an AS. In this case, upon reception of an RSVP Path message requesting the establishment of an LSP, an AS Border Router (ASBR) sends a Path Computation Request (PCReq) to its PCE. This message at least contains information about the head-end of the segment to compute, the tail-end of the LSP and the constraints to be respected by the segment. After the completion of the computation, the PCE replies with a Path Computation Reply (PCRep) message. This message contains a path segment from the ingress ASBR to a BGP Next-Hop (NH) or indicates that there is no path segment respecting the constraints. If the PCE is collocated with the ASBR there is no need for PCReq and PCRep messages. The path segment is computed by the ASBR.

The ASBRs store the list of NHs that have already been tried for an LSP and lead to an infeasible path with regard to the constraints. When the PCE is not able to complete the path with a segment respecting the constraints, “crankback” is performed [FSI+05]. That is, the ASBR generates an RSVP Path Error message and sends it upstream. The upstream ASBR requests from its PCE the computation of a new segment avoiding the NHs that have already been tried.

The role of crankback is crucial for the establishment of interdomain LSPs because only limited information is available concerning the paths to reach a destination outside an AS. Thus, a PCE that computes a portion of a constrained interdomain LSP must rely on heuristics to choose an appropriate BGP NH among the NHs announced for the destination. If a bad choice is performed by the heuristic at some PCE, a downstream PCE may not be able to complete the computation of the path. Crankback enables to cope with such a situation and subsequently try alternative NHs.

Figure 4.3 illustrates the RSVP-TE ERO expansion technique with path computation that takes place inside PCEs. In this example, an LSP with delay constraint of 100 ms has to be established from S to D. Therefore, the source of the LSP S sends a PCReq to its PCE. Inside this message (a), the source specifies the tail-end and the constraints of the LSP. The PCE, $PCE1$, computes a path segment respecting the constraints based on its knowledge of the internal topology and the BGP routes for the destination [3]. Then, it replies with a PCRep message (a) that includes the computed path segment. Upon reception of this reply, S generates an RSVP Path message with ERO object that contains the path segment received from the PCE. The Path message is sent along the segment leading to the establishment of the LSP along the path segment.



Prefix	NH	AS-path
D/16	R21	AS2-AS4
D/16	R23	AS2-AS4
D/16	R31	AS3-AS4

Prefix	NH	AS-path
D/16	R41	AS4

PCReq	Dest	Max delay	Avoid	...
(a)	D	100	-	...
(b)	D	100	R21	...
(c)	D	100	R21 & R23	...

PCRep	Path segment	Delay
(a)	S-R11-R12-R21	3
(b)	S-R14-R23	6
(c)	S-R14-R31	13

Prefix	NH	AS-path
-	-	-

Prefix	NH	AS-path
D/16	R41	AS4

Figure 4.3: ERO expansion

At the ingress ASBR inside $AS2$, $R21$, the process described in the previous paragraph is repeated. That is, $R21$ sends a PCReq to $PCE2$. However, $PCE2$ is not able to provide a path segment that respects the constraints. The only way, known by $PCE2$, to reach D is via the NH $R41$. However, this requires to use

[3]We note that each PCE in fi gure 4.3 knows all the BGP routes that are learned inside the AS, as suggested in chapter 5.

link $R23 - R41$. Since this link has a longer delay than the delay constraint for the remaining LSP's segment, $PCE2$ cannot return a path segment that respects the constraint to $R21$. Thus, $PCE2$ sends back to $R21$ a PCRep that indicates this situation. Consequently, crankback occurs at $R21$.

$R21$ sends a Path Error message upstream. When the Path Error message arrives at S, S sends a new PCReq to $PCE1$. This request (b) contains in addition to the constraints in PCRep (a), the address of the NH that lead to an infeasible path, $R21$. $PCE1$ provides a path segment that ends at NH $R23$. This NH is also a bad choice. S sends a Path message along the path segment toward $R23$ and is notified of the failure to continue the establishment of the LSP after crankback occurs at $R23$ because there is no BGP NH, known by $PCE2$, reachable with a segment that respects the specified delay constraint.

Upon reception of the Path Error message, S sends PCReq (c) to $PCE1$ with the constraint to avoid both $R21$ and $R23$. $PCE1$ replies with path segment ending at NH $R31$ in PCRep (c). A Path message is sent by S along the computed segment. The ingress ASBR, $R31$, in the downstream AS, $AS3$, asks its PCE, $PCE3$, for the computation of a path segment starting at $R31$ and ending at the entrance inside a downstream AS. $PCE3$ replies with path segment $R31-R32-R41$. This path segment is inserted inside the ERO of the Path message and the establishment of the LSP continues until the LSP's tail-end is reached.

Our implementation of this path computation technique differs from the technique proposed in [VAZ06] in the following points. First, we do not consider that the list of domains to be crossed by an LSP is known prior to the computation of an LSP. Second, we assume that the selection of a suitable node to leave a domain and the computation of the path segment inside a domain may be delegated to a PCE by the nodes that are on the path of the LSP, when they need to perform such actions. Finally, in our implementation the inter-AS LSPs are not allowed to cross the same AS multiple times. Thus, it is not possible with our implementation to compute an interdomain path for an LSP with head-end and tail-end nodes in the same AS. A more detailed description of these differences and their implications can be found in section A.3.3 of appendix A.

4.2.4 Cooperative PCEs

A cooperative PCE [FVA06] communicates with other PCEs in order to request or delegate the computation of path segments contained in regions for which it does not possess enough topological information.

As mentioned earlier, a cooperative PCE asking for a path computation from another PCE is considered as a PCC. Thus, the communication protocol used between cooperative PCEs is the protocol defined in [VLA^{+}06] for communication between PCCs and PCEs.

In this technique, a PCReq message specifying the constraints for the LSP is sent from the PCE of one domain to the PCE of the downstream domains. Upon reception from the PCEs in the downstream domains, of multiple path segments starting at the entrances of the downstream domains and ending at the LSP's tail-end together with of their QoS properties, a PCE is capable of computing the best segments starting at the entrances of its domain and ending at the tail-end of the LSP, with regard to the constraints. These segments are sent to the upstream PCE inside a PCRep message.

This path computation technique, called Backward Recursive PCE-based Computation (BRPC), is described in [VZB06]. It has been designed to find the shortest path for a constrained inter-AS LSP request [4]. It makes the assumption that the list of domains to be crossed by the LSP is known prior to the computation. Thus, the computed path is the best path that can be obtained along this interdomain path.

The PCE working group of the IETF is mostly focussed on inter-area TE LSPs. In this case, the areas to be crossed by the LSPs are known prior to the computation of the path because an AS is usually divided in a backbone area and stub areas connected to the backbone. The path between two stub areas goes from the area of the head-end to the backbone area and, finally, ends in the area of the tail-end. Thus, the assumption concerning the knowledge of the domains to be crossed by the LSPs holds for inter-area LSPs, if each domain corresponds to an area. However, the ASs to be crossed by a constrained inter-AS LSP cannot be known in advance. The `IPsphere FORUM` [IPs] is developing a solution to determine the ASs to cross for an LSP with given QoS requirements and taking into account business relationships of the ASs [5]. The list of ASs to be crossed by a QoS constrained LSP could also be given by QoS extensions to BGP. This would require to advertise one route for each QoS class with BGP, leading to an increase in BGP routing tables size and number/size of update messages exchanged.

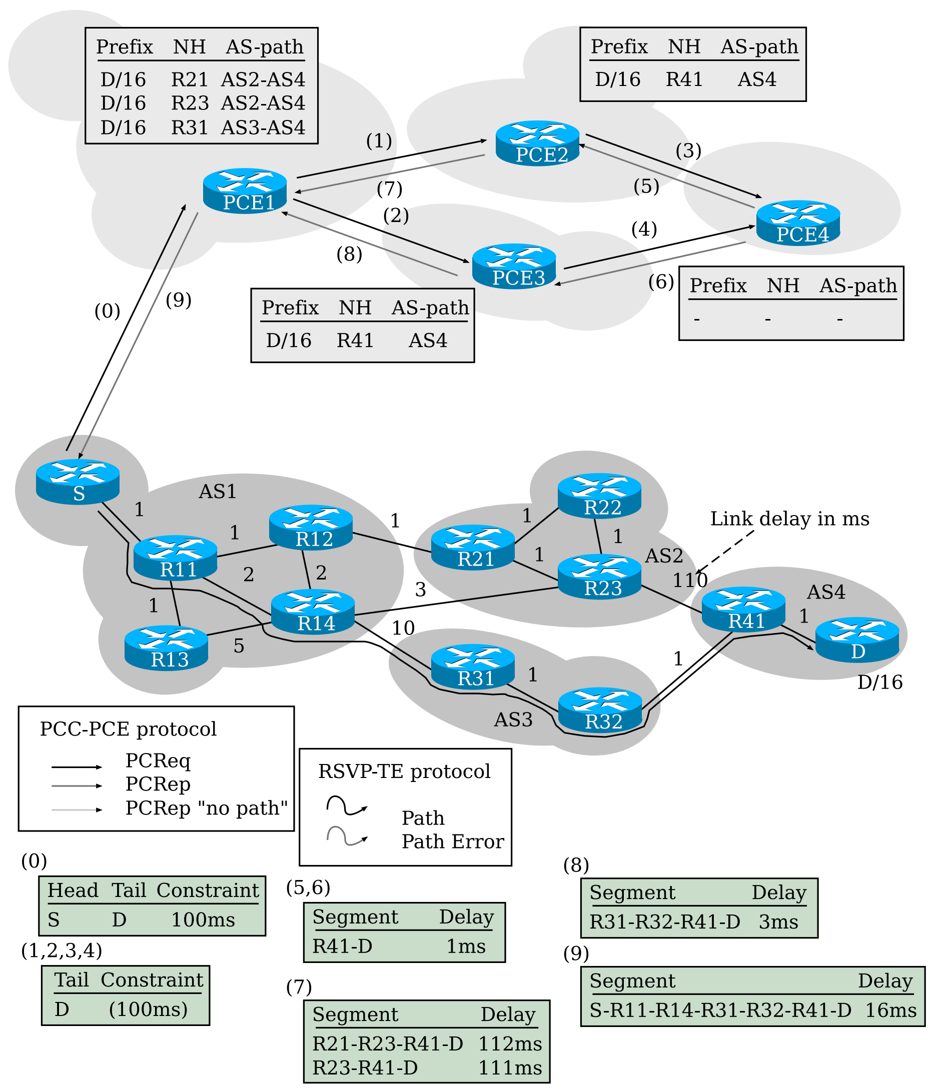
Figure 4.4 illustrates the computation of inter-AS constrained paths by means of cooperative PCEs. We assume that the list of domains to be crossed by the LSP is not known a priori contrary to [VZB06]. Thus, the PCEs contact a PCE in each downstream domain available from the TED, in order to find the shortest path for the LSP. As a consequence, the optimality of the path found for the LSP depends on the content of TED. Here, we assume that the TED of a PCE is composed of all the BGP routes learned inside the AS[6]. In general, the TED could be populated by other means than with BGP. However, a PCE has to be able to determine a set of possible downstream domains from the TED and to contact the PCEs that are discovered inside these domains. The LSP to establish is subject to a maximum delay constraint of $100ms$. The head-end of this LSP is router S in $AS1$. The tail-end of the LSP, node D, belongs to $AS4$. The longest matching prefix advertised for D is $D/16$. The central part of the figure shows the physical topology of the ASs and their interconnections. In the top part of the figure, we see the PCEs of

[4]The term shortest path applies to a path with smallest end-to-end value for an additive metric of the links such as the delay or the cost.

[5]Participation to the `IPsphere FORUM` is based on membership. Their work is not publicly available.

[6]The reason for the collection at the PCE of all the BGP routes that are learned inside the AS is provided in chapter 5.

each ASs, labels for the messages exchanged between PCEs and the BGP routes known by the PCEs. The content of the messages exchanged between PCEs is shown at the bottom of the figure.



Prefix	NH	AS-path
D/16	R21	AS2-AS4
D/16	R23	AS2-AS4
D/16	R31	AS3-AS4

Prefix	NH	AS-path
D/16	R41	AS4

Prefix	NH	AS-path
D/16	R41	AS4

Prefix	NH	AS-path
-	-	-

(0)

Head	Tail	Constraint
S	D	100ms

(1,2,3,4)

Tail	Constraint
D	(100ms)

(5,6)

Segment	Delay
R41-D	1ms

(7)

Segment	Delay
R21-R23-R41-D	112ms
R23-R41-D	111ms

(8)

Segment	Delay
R31-R32-R41-D	3ms

(9)

Segment	Delay
S-R11-R14-R31-R32-R41-D	16ms

Figure 4.4: Cooperative PCEs - BRPC

The head-end of the LSP, S sends a PCReq message to the PCE of its AS, $PCE1$. $PCE1$ in $AS1$ has three routes for prefix $D/16$. We observe from the `AS-path` that two of these routes are received from $AS2$, with two different BGP Next-Hops (NHs) and the other route is received from $AS3$. Thus, it sends a PCReq message to $PCE2$ and $PCE3$, the PCEs inside $AS2$ and $AS3$. The PCReq messages contain the address of the tail-end of the LSP and the constraints for the

LSP. The delay constraint is not necessary because the output of the computation technique is the shortest delay path. If the delay of the path returned to the LSPs head-end is above the delay constraint, there is no suitable path for the LSP.

$PCE2$ and $PCE3$ have one route for prefix $D/16$. The NH for this route is router $R41$ in $AS4$. Therefore, $PCE2$ and $PCE3$ both send a PCReq to $PCE4$. $PCE4$ computes a path segment from $R41$ to D, the tail-end of the LSP. Then, it sends the segment with its delay in PCRep messages (5) and (6) upstream to $PCE2$ and $PCE3$, respectively.

When $PCE2$ receives PCRep (5), it has the response to the single PCReq that it sent for the LSP. It has all the requested information. Thus, it is now able to compute path segments from the entrances inside its domain to the destination of the LSP or to determine that is such path segments respecting the requested constraints cannot be provided. For this purpose, the PCE performs a SPF computation[7] on the graph composed of the local topology, the inter-AS links and the segments received from the downstream PCE. The result is two path segments starting at nodes $R21$ and $R23$, ending at D. Upon reception of PCRep (6), $PCE3$ performs the same actions as described for $PCE2$.

Next, $PCE2$ sends the resulting segments and their delays inside PCRep (7) to $PCE1$. In addition, $PCE3$ sends path segment from $R31$ to D and the delay of the segment to $PCE1$, in PCRep (8). Because $PCE1$ received replies from all the PCEs it sent PCReq messages to, it computes the end-to-end path based on the local topology, the inter-AS links connected to $AS1$ and the received segments. The resulting path is $S - R11 - R14 - R31 - R32 - R41 - D$ with delay of $16ms$. This path is sent in PCRep (9) to the head-end of the LSP, S. Finally, S initiates the establishment of the LSP along this path. For this purpose it stores the list of nodes along the computed path inside the ERO. Thus, the RSVP Path message follows the computed path and the LSP is established along this path.

In order to respect the confidentiality requirement of SPs (see section 3.4), PCEs may return path keys [BVF06] inside PCRep messages, instead of returning path segments that reveal sequences of hops inside their domains. A Path Key Sub-object consists of a key that replaces the Confidential Path Segment (CPS) generally contained inside the domain of the PCE and of the identifier of the PCE. Path Key Sub-objects (PKS) can be stored inside the ERO of the RSVP Path messages. Such sub-object must follow the node responsible for expanding the path key, that is the first node of the confidential path segment. This node sends the path key to the PCE with identifier contained in the PKS for expansion of the path key into a sequence of nodes.

In section A.4 of appendix A, we provide a detailed description of our implementation of this technique. We implemented this technique in parallel to the elaboration of BRPC [VZB06] at the IETF. Thus, our implementation differs from BRPC in matters that were not described at the IETF at the time of our implementation. Moreover, our implementation solves certain issues that have not yet been

[7]Another algorithm could be used.

considered at the IETF. We show in appendix A that these divergences do not have an impact on the resulting computed paths. However, they induce differences on the information and number of messages exchanged between cooperative PCEs. In the next paragraphs, we give a brief overview of these differences. A more detailed description is provided in section A.4.2 of appendix A.

The first difference is that, in our implementation, we do not assume that the list of domains to be crossed by an LSP is known prior to the computation of its paths. It results that all the PCEs that are downstream of the current PCE on the path to the LSP's tail-end need to be contacted if the optimal path to the destination has to be found. This difference implies that a PCE may receive multiple requests (PCReqs) for the same LSP. These requests may have crossed a different list of upstream PCEs. Our implementation has to distinguish such PCReqs from PCReqs that are looping. A looping PCReq results in the generation of a Path Computation Error (PCErr) message. The other PCReqs should trigger a PCReq message as reply.

Because PCEs may receive multiple requests for the same LSP, they may benefit from maintaining a cache in order to avoid recomputations and to reduce the exchange of messages with other PCEs. In our implementation, we distinguish two types of PCEs. Stateless PCEs do not maintain a cache with the computed path segments and the responses from the downstream PCEs contacted before, for an LSP. These PCEs need to recompute the local paths segments and to contact all the downstream PCEs each time a PCReq is received. On the contrary, stateful PCEs maintain a cache for each LSP. This cache contains the local paths computed for the LSP and the responses from the downstream PCEs that were contacted for the previous requests. A stateful PCE does not need to recompute already known local path segments. Moreover, it only sends PCReq messages to the PCEs that did not participate in previous computations for the available NHs.

The second difference between BRPC and our implementation concerns the computation of the local path segments. In BRPC, a PCE computes the segments from its ingress ASBRs to a set of ingress ASBRs inside the downstream domains upon reception of the replies to the PCReqs that it sent. In our implementation, the local path segments are computed on receipt of a PCReq message. We show in section A.4.2 of appendix A that this difference has an impact on the number of messages exchanged during the computation of interdomain paths. In certain cases, BRPC requires fewer messages to be exchanged while in others fewer messages are exchanged with the technique that we implemented.

In section A.4, we describe the algorithm that we use for the computation of the lower and upper bounds on the number of messages that are exchanged in the computation of each LSP with our implementation of cooperative PCEs. The number of messages exchanged for the computation of a particular LSP with our implementation of cooperative PCEs is comprised between these bounds. This number approaches the upper bound if the PCEs are stateless. The number of messages exchanged between cooperative PCEs approaches the lower bound if all PCEs are stateful and maintain the content of their cache during an infinite period of time. The lower bound is also a lower bound on the number of messages exchanged with BRPC. However, we show in A.4.2 that the upper bound on the number of messages with BRPC may be higher than the upper bound computed for our implementation with algorithm 4.

4.2.5 Comparison

Table 4.1 provides a synthetic comparison concerning the properties of the path computation techniques described in section 4.2. Each line in the table is related to a different property. Each column is relative to a path computation technique.

	Technique			
	IP forw.	Global PCE	ERO exp.	Coop. PCEs
Simultaneous path comp.	-	Yes	No	Yes
Shortest path comp.	-	Yes	No	Yes
Inter-domain PCC-PCE communication	No	No	No	yes
Intra-domain PCC-PCE communication	No	Yes	Yes	Yes
Head-end-PCE communication	No	Yes	Yes	Yes
ASBR-PCE communication	No	No	Yes	No

Table 4.1: Comparison of the path computation techniques

The "simultaneous path computation" property concerns the capability of computing a protected LSP and its disjoint backup LSP at the same time. This enables to find a pair of disjoint paths to the destination if such paths exists in the topology. Algorithms that compute the protected and backup paths consecutively may not be able find a path for the backup LSP. They are subject to the so-called trap topology problem. The "simultaneous path computation" criteria does not apply to IP forwarding because it is not possible to make sure that a router possesses two disjoint IP forwarding paths for a destination. A global PCE has the complete topology information of the ASs that can be crossed by the protected and its backup LSP. It has enough information to use an algorithm, such as Bhandari's algorithm [Bha99], solving the min-cost disjoint path pair problem instead of CSPF in order to compute the protected and backup LSPs simultaneously. The ERO expansion technique does not perform a simultaneous path computation of the disjoint pair of LSPs. It is not possible to avoid the trap topology problem with ERO expansion because the downstream ASBRs are selected without knowledge of the available paths downstream of the ASBRs. Choosing an ASBR for the first LSP does not ensure that another suitable ASBR will be found for the backup LSP. The cooperative PCEs path computation technique described in [VZB06] is capable of computing both protected and backup LSPs at the same time.

The second criteria in table 4.1 concerns the capability of the computation technique to return the shortest path available from the source to the destination node. The IP forwarding path between two nodes in different ASs is determined by other criteria in addition to the cost metric of the links along the path. Thus, IP forwarding paths may not be the shortest paths available in the topology. However, the global PCE and cooperative PCEs are able to compute the shortest path between two nodes with respect to a given metric. For cooperative PCEs to return the shortest path, each participating PCE has to contact the PCEs in all the ASs connected to its domain. ERO expansion does not compute the shortest path but it can return a path with cost that is below a given value.

Now, we consider the communication implying PCEs. There is no PCE involved in the computation of IP forwarding paths. Thus, there is no exchange of messages with PCEs in IP forwarding path computations. We see in line 3 of table 4.1 that only cooperative PCEs require PCReq and PCRep messages to cross AS boundaries. Thus, it may be required to filter these messages at ASBRs or PCEs to apply AS policies. With ERO expansion, policies are applied at ASBRs at the establishment of LSPs, upon reception of Path messages.

We note in table 4.1 that all techniques except IP forwarding require intra-domain PCC-PCE communication. With the global PCE, ERO expansion and cooperative PCEs, there is communication between the head-end LSR and the global PCE. In addition, there is also ASBR-PCE communication in the ERO expansion technique.

4.3 Related Work

Up to now there has been very few research focussed on providing QoS services across AS boundaries while the provision of QoS inside a domain is widely addressed in the literature.

Ho et al. [HWPTH04] consider the availability of QoS extensions to BGP for the selection of an egress router to provide services with bandwidth guarantees. QoS extensions to BGP are proposed by Jacquenet [Jac03] and Boucadair [Bou05]. These extensions enable to advertise the QoS of the interdomain routes. These extensions have however not been evaluated nor deployed, even though they are likely to generate a lot of signaling messages due to the dynamics of the QoS information.

In [YSLMB$^+$05] and [FBR$^+$04], the authors define an architecture with a centralized entity inside each domain. They propose to define a new interdomain routing protocol to be used between the entities, either called Inter-Domain Routing Agent or Routing Control Platform. They propose to exchange QoS information with this routing protocol. A mechanism for the negotiation of Service Level Specification (SLS) is also defined in [H$^+$05] as a support for the provision of QoS-based services.

[OHMC01] develops an architecture for the provision of Differentiated Services that is similar to the cooperative PCEs architecture proposed in [FVA06] and evaluated in the following chapters. Both [OHMC01] and [FVA06] make the assumption of a method to select the downstream domain to be contacted by the path computation entity. In this paper, we do not rely on such mechanism. In our evaluation of the cooperative PCEs architecture, all the PCEs inside downstream ASs obtained from the BGP routes are contacted. The IPsphere forum [IPs] is working on a solution to determine the list of downstream domains that are able to support a given service. They propose the introduction of a business layer for this purpose. The list of downstream domains may also be determined based on aggregated routing information distributed between PCEs [YLT$^+$06]. In case the list of domains to be crossed by an LSP is not known, Torab et al. [TJX$^+$06] propose that a cooperative PCE sends Path Computation Requests to downstream PCEs in an analogous way as packets are forwarded in ad-hoc routing protocols. Unfortunately, the authors do not evaluate such a proposal.

4.4 Conclusion

In this chapter, we have first described the role of Path Computation Elements (PCEs) in the computation of constrained paths. We have defined the notions related to computation with PCEs, the communication with PCEs and the mechanisms required for the use of PCEs.

Then, we have presented four different path computation techniques. The first technique is the classical IP forwarding. We have shown that IP forwarding does not enable to find paths respecting given QoS properties. The second technique is called global PCE. It assumes the knowledge of the complete topology of different ASs at a single node, the global PCE. We have seen that this technique is not scalable and not applicable for the computation of LSPs crossing ASs that do not want to reveal their internal topology to their peers. These two techniques are used for comparison purposes with the two other path computation techniques.

The other two techniques are distributed. One of these techniques is called ERO expansion. We have seen that it may or may not rely on PCEs. In this technique, the path computation and the LSP establishment take place at the same time. The computation ends once a path respecting the constraints is found even if it is not the shortest path. With this technique it cannot be ensured that a pair of disjoint path will be found in case of a trap topology. However, according to [Gro04], trap topologies are not frequent in communication networks. With ERO expansion, if PCEs are used in the computation, they do not communicate among themselves. Thus, discovery of PCEs in neighboring domains is not required.

The last path computation technique relies on PCEs that communicate and cooperate in order to find the shortest path respecting given QoS constraints. We call this technique cooperative PCEs. Here, PCEs have to discover the PCEs in neighboring domains. If the list of domains to be crossed by the LSP is not known prior to the computation, a path is computed along all the possible AS-paths toward the destination. Only the shortest path is selected for the establishment of the LSP. Path computation takes place before LSP establishment. With this technique, it is possible to simultaneously compute a pair of disjoint LSPs, as described in [VZB06], when the AS-path for the pair of LSPs is given.

In this chapter, we have described the current state of development at the IETF of the ERO expansion and cooperative PCEs techniques. We have provided a theoretical comparison of these techniques that are evaluated in chapters 5, 6 and 7. The difficulties in implementing these techniques and the different choices that we made for the implementation are introduced in appendix A. In appendix A, we show that our implementation choices produce the same paths as the techniques presented in this chapter. However, we draw attention on the fact that the number of messages exchanged in the computation of the paths, with cooperative PCEs, and the information carried in the messages is different from the exchange of messages between PCEs in the BRPC [VZB06] technique.

Chapter 5

Path Quality and TED Content

When presenting distributed path computation techniques relying on PCEs in the previous chapter, we said that the quality of the paths computed by these techniques depends on the content of the Traffic Engineering Database (TED) used for the computation. The aim of this chapter is to quantify the impact of the TED content on the quality of the computed paths and the amount of established LSPs. We consider that the TED is populated with BGP routing information and the topology of the local AS distributed by the IGP. Our objective in this chapter is to determine which nodes in an AS are suitable for the computation of constrained interdomain paths. We want to determine if nodes along an inter-AS LSP are able to complete the path computation for the LSP in the ERO expansion technique as suggested in [VAZ06]. In this chapter, we study the capability of the nodes to become PCEs based on the routing information available at the nodes.

In the short term, we expect that the first motivation for using MPLS across interdomain boundaries will be to provide services such as multi-AS VPNs or to interconnect large telephone switches in different domains (VoIP traffic). We expect that those services will initially be deployed between ASs that are directly connected and likely managed by the same company [CC04]. Thus, in this chapter, we consider two multiply interconnected ASs.

The case of two interconnected ASs is the simplest case that may require the distribution of routes with BGP. In this chapter, we first describe the ways to distribute the BGP routes inside an AS, iBGP full-mesh and Route Reflectors (RRs). We discuss their implications on the routes available at each router. We see that there may be a loss of routing information toward a prefix at each BGP router that readvertises the prefix. Then, we study, by means of simulations, the computation of diverse paths based on BGP routes between two neighboring ASs. We present, in section 5.2 the particularities of the computation techniques used in the simulations. In section 5.3, we provide the settings of our simulations. We consider different locations for the PCEs. We compare the distributed computation of LSPs by PCEs that are located inside Provider Edge (PE) routers and ASBRs to the computation performed by PCEs located inside RRs. In the first case, the computation is performed by the nodes that belong to the LSP's path, as proposed in [VAZ06]. In the latter case, the RRs perform the computation on behalf of their clients. These RR-clients belong to the LSP's path. In section 5.4, we provide recommendations on the location of the PCE and on the content of its TED based on our observations from section 5.3 and appendix B. Finally, we conclude the chapter.

5.1 BGP Route Distribution

The routing information distributed by BGP is very different from the topology information distributed by IGPs such as OSPF or IS-IS. BGP [Ste99] is a path vector routing protocol. It is much more scalable than a link-state IGP in that it only distributes reachability information subject to routing policies that limit the routes announced to neighboring ASs. The price for this scalability is the lack of information available on the Internet topology [CGJ$^+$04]. For each prefix, each peer only advertises its best route over BGP sessions. This route is selected based on criteria that are independent of the quality of the route in terms of end-to-end metrics like delay and reservable bandwidth as mentioned in chapter 2. When considering VPN services across domains or, more generally, LSPs crossing multiple domains, the limited topological information available through BGP interdomain routes becomes a problem.

In figure 5.1, we see the BGP sessions that are required to distribute BGP routing information with a full-mesh of iBGP sessions inside each AS. On top of the figure, we have the physical topology. At the bottom of the figure, we have the BGP sessions. A router attached to another AS via a peering link establishes an eBGP session over the peering link with the BGP neighboring router. For example, in figure 5.1 there are three eBGP sessions between $AS1$ and $AS2$. There are eBGP sessions between R13 and R21, R13 and R22, and also between R14 and R22. The eBGP sessions are used to advertise the routes that are reachable by each AS. A BGP router advertises its best route to reach each destination prefix. When a BGP router receives a route over an eBGP session, it determines whether this route is its best route towards the destination. If so, it advertises the route to the other BGP routers of the AS. This is done by means of iBGP sessions. We note, in figure 5.1 that there are 10 iBGP sessions in $AS1$.

BGP was initially designed assuming a full mesh of iBGP sessions between all the border routers of an AS. The distribution of the BGP routes inside an AS is illustrated in figure 5.2. With a full mesh of iBGP sessions, the best eBGP routes selected by the border routers, routes (3) and (4) in figure 5.2, are distributed in the AS. This allows each router to compute its best route towards any reachable destination. Due to this assumption of a full-mesh of iBGP sessions, a BGP router does not advertise, over iBGP sessions, a route received over an iBGP session. The other routers in the iBGP full-mesh have also received this route. Thus, if a router selects a route received via iBGP as best route, it will not advertise routes learned

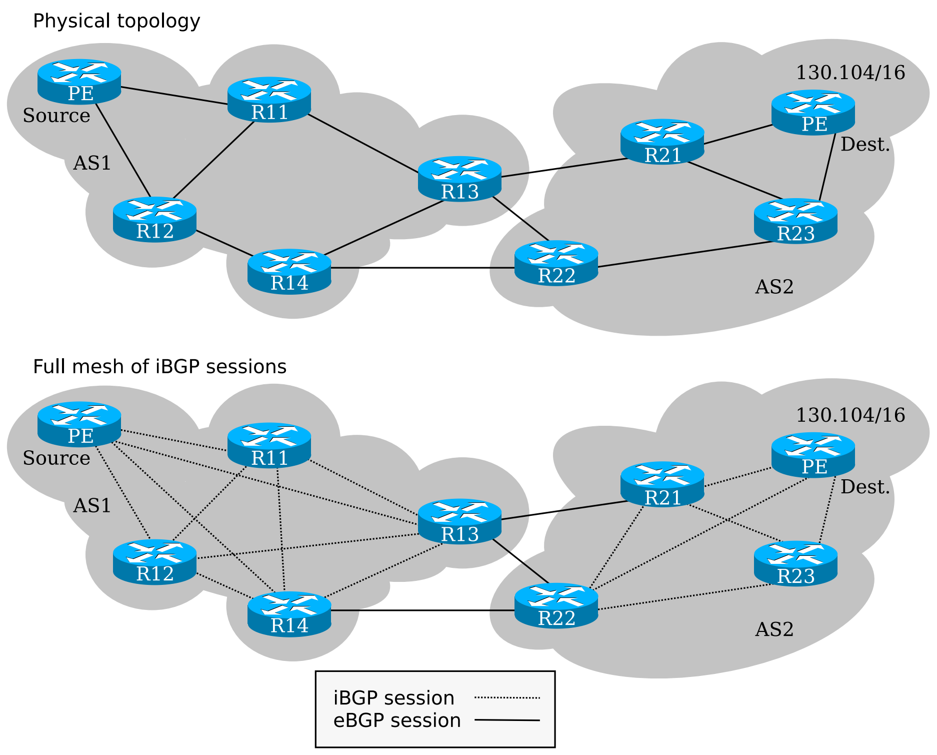


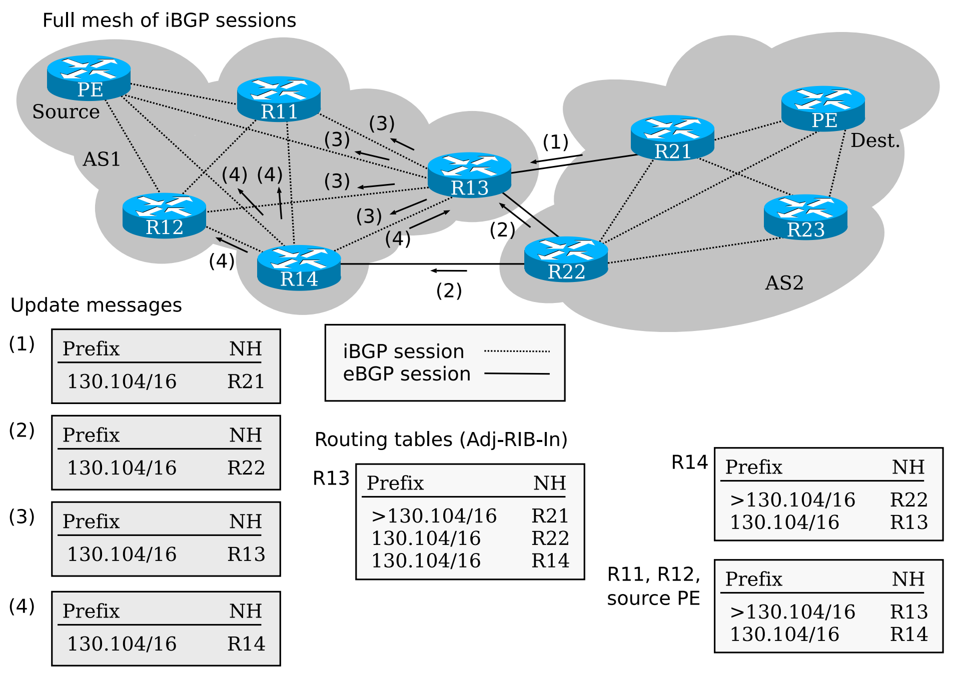
Figure 5.1: Full mesh of iBGP sessions

on eBGP sessions inside the AS[1]. The reason is that a BGP router only advertises its best route. Moreover, a router that has multiple eBGP sessions advertises at most one of the routes learned on eBGP sessions on its iBGP sessions. For example, $R13$ in figure 5.2 advertises a single route for prefix $130.104/16$ even if it has to ways to join the prefix. Thus, the routes learned on the other eBGP sessions are not distributed inside the AS. As a consequence to the BGP route selection and distribution mechanisms, there may be many available interdomain paths that are never learned by the routers and thus never used for packet forwarding.

If there are N border routers in the AS, a full mesh of iBGP sessions corresponds to $\frac{N \times (N-1)}{2}$ iBGP sessions. This is a severe scalability problem in networks containing more than a few tens of border routers. Two solutions have been proposed to solve this problem : confederations [Tra96] and route reflectors (RRs) [BCC00]. We do not consider the confederations in this thesis as they are not frequently used.

A route reflector is a router that is allowed to re-advertise, over iBGP sessions, routes that it received over other iBGP sessions. The simplest way of deploying

[1]This case is not illustrated in the fi gure.



Update messages

(1)

Prefix	NH
130.104/16	R21

(2)

Prefix	NH
130.104/16	R22

(3)

Prefix	NH
130.104/16	R13

(4)

Prefix	NH
130.104/16	R14

Routing tables (Adj-RIB-In)

R13

Prefix	NH
>130.104/16	R21
130.104/16	R22
130.104/16	R14

R14

Prefix	NH
>130.104/16	R22
130.104/16	R13

R11, R12, source PE

Prefix	NH
>130.104/16	R13
130.104/16	R14

Figure 5.2: Route distribution with a full mesh of iBGP sessions

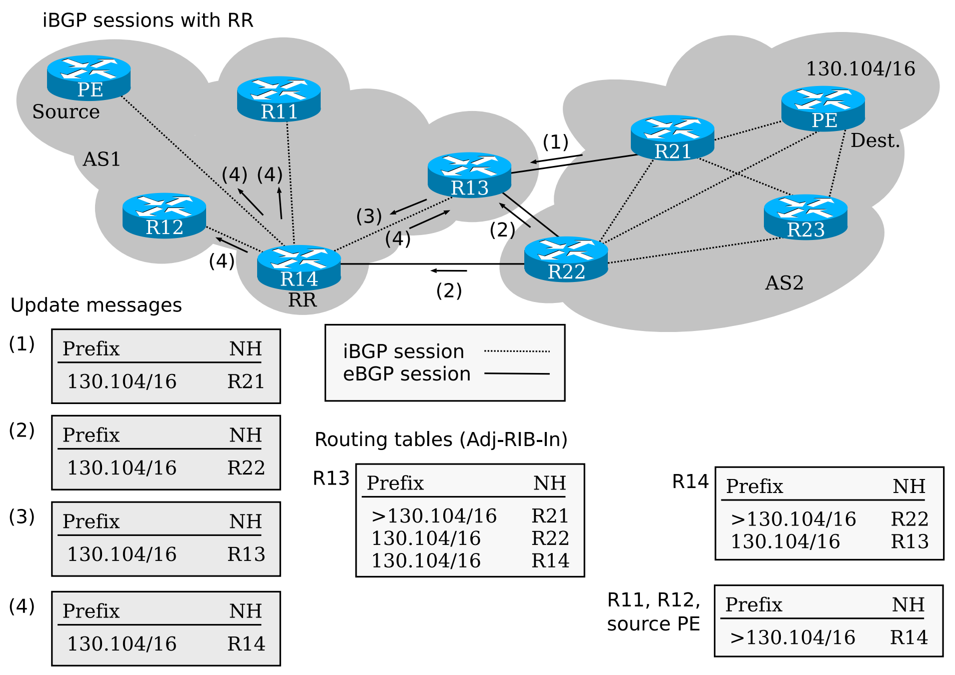
RRs is to replace a full mesh of iBGP sessions with a single RR. This is illustrated in figure 5.3. When a single RR is connected to all other BGP routers of the domain, each BGP router receives only one route from the RR instead of the $N - 1$ routes received in the case of a full mesh of iBGP sessions. We observe in figure 5.3 that the source PE, $R11$ and $R12$ only learn one route from their RR, $R14$, for prefix $130.104/16$ instead of two routes in figure 5.2. Thus, the introduction of RRs may further reduce the number of routes available at RR-clients.

Moreover, The placement of RRs inside a domain might create convergence problems and forwarding loops [GW02b, BOR$^+$02]. These can be avoided by following the recommendations of [BCC06]. Vutukuru et al. [VVKB06] also proposed a way to place RRs and configure iBGP sessions in order to avoid these problems.

5.2 Path Computation Techniques

In order to determine a suitable location for the PCE functionality, we consider two alternative techniques. The centralized path computation technique presented in section 4.2.2 and the distributed computation technique called ERO expansion, presented in section 4.2.3.

The first technique is used as a benchmark. It consists of a centralized approach where the node possessing the intradomain topology of all the ASs that



(1)

Prefix	NH
130.104/16	R21

(2)

Prefix	NH
130.104/16	R22

(3)

Prefix	NH
130.104/16	R13

(4)

Prefix	NH
130.104/16	R14

R13

Prefix	NH
>130.104/16	R21
130.104/16	R22
130.104/16	R14

R14

Prefix	NH
>130.104/16	R22
130.104/16	R13

R11, R12, source PE

Prefix	NH
>130.104/16	R14

Figure 5.3: Route distribution with a RR

can be crossed is responsible for the computation of the interdomain paths. This technique is only applicable for LSPs crossing a few ASs belonging to the same company. The second approach is applicable in a more general framework. It is a decentralized technique where each node on the path of the LSP completes the path computation toward the destination based on local routing information. This technique is applicable for the establishment of LSPs crossing any number of ASs. We have proposed in section 4.2.3 that the computation be performed at PCEs instead of at nodes along the LSP. In this chapter, we use the ERO expansion technique to evaluate the gain of using PCEs compared to the computation at nodes that belong to the LSP. In addition, we determine, based on simulations with this technique, the content of the TED that is required at the PCEs to be able compute constrained paths on behalf of other nodes.

The LSPs considered in this chapter are subject to maximum end-to-end delay and bandwidth guarantees as well as link and node disjointness constraints, for the backup LSPs. The backup LSPs are subject to the same maximum end-to-end delay constraint and bandwidth reservation as the LSPs they protects. The LSPs cross two directly connected ASs. They connect PE routers in the neighboring ASs.

5.2.1 Centralized Path Computation with CSPF

A centralized path computation can only be considered for LSPs crossing ASs that belong to the same company as ISP topology information is often considered strategic and kept secret [SMWA04]. In that case, a global PCE that centralizes the topology information of multiple ASs can compute the path for inter-AS LSPs remaining inside this set of ASs.

The PCE collects the link state packets advertised by the IGP in the ASs and thus possesses the complete topology of the ASs with the TE information, if either IS-IS TE or OSPF-TE is used. For the purpose of this thesis we assume that both delay and reservable bandwidth are distributed by the IGP[2]. Based on this information, the PCE runs a CSPF algorithm. It prunes the links with insufficient remaining reservable bandwidth, runs the Dijkstra algorithm with costs set to the delay of the links and finally sends the computed path to the source of the LSP, if the path respects the delay constraint. For the disjoint path computation, the PCE first prunes the links and nodes that are on the primary path from the topology with the exception of the LSP's head and tail-end. Then, it runs the computation as for the primary path.

5.2.2 BGP-based Distributed Path Computation (DPC)

Distributed interdomain path computation techniques have to be considered, because it may not be possible or desirable that a single node knows the complete intradomain topologies of several ASs.

The distributed computation technique considered in this chapter is ERO expansion. Computation of the paths with ERO expansion is faster than with cooperative PCEs. Additionally, since the trapping problem is not frequent in communication networks [Gro04], both techniques should find a suitable pair of disjoint paths for the same LSPs, in the simulations without bandwidth reservations.

In ERO expansion, a node that needs to complete the path for an LSP because the first node in the ERO of the Path message is a loose hop that is not directly connected or an abstract node, relies on the routing information distributed by BGP. Each router uses a single best BGP route to forward IP packets toward each distant destination prefix. These routes are stored in its Local Routing Information Base (Loc-RIB). However, a router may receive one route toward each prefix from each of its peers. If they pass the import filters, these routes are stored in its Adj-RIB-Ins. We use these routes to compute our constrained paths.

We assume the use of next-hop self, in this chapter. However, in our simulations, the use of the option does not have an impact on the diversity of the paths that can be computed in a distributed manner (see appendix B for more details). With this option, a BGP router replaces the NH of a route by its own IP address before readvertising the route inside the AS. This option is commonly used because it

[2]We assume that the IGP cost of a link is set to its propagation delay. The TE extensions to OSPF and IS-IS enable to advertise the unreserved bandwidth of a link.

avoids having to advertise the peering routers of neighboring ASs inside the IGP of the AS. However, it results in more nodes having to expand the ERO. We invite the reader to go to section B for a discussion on the impact of the use of the next-hop self option in the distributed computation of the inter-AS constrained paths.

The distributed computation of a protected LSP is illustrated in figure 5.4. Inside the source AS $AS1$, the source (PE) router selects, from all the routes toward the destination PE present in its Adj-RIB-Ins, the route with the Next-Hop (NH) that is reachable through a path with enough reservable bandwidth and the smallest delay. This consists in performing a CSPF inside the source AS toward all the NHs that advertise the destination prefix, with the delay as metric. Once the NH $R13$ is selected, the LSP is established toward this NH using RSVP-TE with an ERO containing the computed constrained path segment $R11 - R13$. The NH $R13$, i.e. the egress ASBR, then selects a NH in the neighboring AS from the NHs of the routes to the destination PE, in the local Adj-RIB-Ins. Therefore, $R13$ evaluates the reservable bandwidth and the delay toward each of these NHs, $R21$ and $R22$. $R14$ is not evaluated to avoid routing loops. Finally, the ingress ASBR $R22$, inside the downstream AS $AS2$, computes the path toward the PE[3] by running a CSPF on the topology of the destination AS.

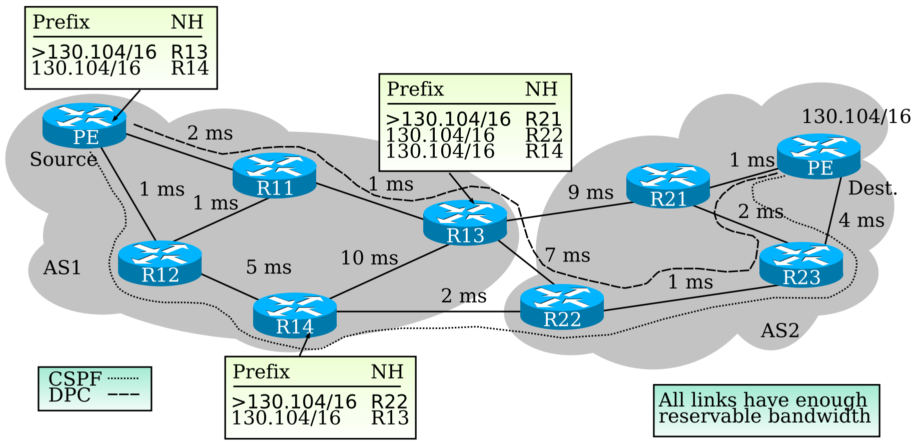


Figure 5.4: Distributed Path Computation of a protected LSP

We observe that the path computed in figure 5.4 has a larger delay than the CSPF path. This is due to the limited information available locally for the route selection. In this distributed computation technique, nodes make a local choice that may not lead to the globally optimal path.

Moreover, in figure 5.4, once the protected LSP is established, an end-to-end link and node disjoint path cannot be found. This comes from the consecutive computation of the protected and its backup LSP. In order to establish a disjoint

[3]If the PE does not belong to this AS, the ingress ASBR selects a NH from the routes in its Adj-RIB-Ins.

path, the nodes that complete the backup path, i.e. the ASBRs in our case, need to know the links and nodes crossed by the protected path. For this purpose, the nodes along the protected path can be recorded in the Record Route Object [ABG$^+$01]. Then, the source of the LSP stores these nodes in the eXclude Route Object (XRO) defined in [LFC05]. This object is used by intermediate nodes to compute path segments that avoid the nodes stored in this object. Based on the XRO, the source PE router selects a NH that does not belong to the primary LSP and that is reachable with a path segment respecting the delay, bandwidth and disjointness constraints ($R14$ in figure 5.4). However, router $R14$ cannot continue the establishment of the disjoint LSP. The two NHs available for prefix $130.104/16$ are already on the path of the protected LSP, hence crankback takes place. A Path Error message is sent to the PE router. The PE router does not possess any other route with a NH that has not already been explored. Thus, the backup LSP cannot be established.

If RRO aggregation is performed inside all ASs, only the border nodes that are crossed by the protected LSP appear as is, in the RRO. The other sub-objects present in the RRO are abstract nodes. However, the RRO is still copied inside the XRO. A node that has to complete the path for the backup LSP contacts a node on the path of the protected LSP to obtain the path segment used by the protected LSP inside its domain. The node to contact is known from the content of the XRO. This procedure is provided in chapter 3. An alternative to RRO aggregation is proposed in [BVF06].

From the discussion in section 5.1 and the example of figure 5.4, we observe that a few elements prevent a straight forward computation of constrained interdomain paths. These elements are:

- The lack of QoS advertisement in BGP
- The lack of topology information concerning the neighboring ASs
- The way BGP routers select and redistribute the BGP routes

Concerning the redistribution of the BGP routes, we have seen in section 5.1, that the redistribution of only the best route to neighboring routers and the introduction of RRs inside ASs have an impact on the number of routes available at nodes participating in distributed constrained path computation. In the next section, we evaluate the impact of the BGP routes available at nodes along the LSP in different iBGP configurations on the computation of a suitable path for the LSP. Then, we consider the computation at PCEs. We evaluate the ability of RRs in filling the PCE functionality of computing constrained interdomain LSPs on behalf of their clients.

5.3 Simulations on TED Content

Our simulation environment contains two ASs. Each AS contains several interconnected routers. Furthermore, the routers in each AS are grouped in POPs as in most networks. A small POP may contain a single router while a large POP may be composed of a few tens of routers. The ASs are interconnected with one peering link in each city where both ASs have a POP. To establish interdomain LSPs, we consider the case of inter-AS VPNs where each AS may offer VPNs services toward the POPs of the other AS. For this reason, we attach a Provider Edge (PE) router to each POP containing more than one router. This PE router is connected to two different routers inside the POP for redundancy reasons. We establish a full mesh of traffic engineered LSPs between those PE routers.

The AS topologies, with link delays and routers grouped in POPs, used for this purpose, have been collected by the rocketfuel project [MSWA02]. We assigned a bandwidth of 10 Gbps to each link. Moreover, each link connecting a PE router to other routers has a delay set to 1 ms. The same delay of 1 ms is assigned to the inter-AS links that we added to interconnect the ASs two by two. A router in each POP is configured as a route reflector, all the routers inside the POP are fully meshed from an iBGP viewpoint, for optimal intra-POP routing, and the route reflectors themselves are fully-meshed as recommended by [BCC06].

In table 5.1, we find the ASs involved in each topology with the number of nodes as well as the number of intra and inter-domain links. The last column indicates the number of LSPs to be established. We note that the number of inter-domain links varies from 3 to 14 links. The topology, “topo3”, with most inter-domain links does not contain the largest number of nodes and links. The biggest topologies in terms of links and nodes are “topo4” and “topo7”. We note that not all the ASs could be interconnected because they did not all have POPs in common locations. [MSWA02] provides 6 AS topologies. From these 6 AS topologies, we generated 8 topologies composed of two interconnected ASs.

Topology	ASs		Nodes	Links			LSPs
	ASN1	ASN2		intra	inter	total	
topo0	3257	3967	281	557	3	560	828
topo1	1239	3967	443	1217	5	1222	1116
topo2	3967	6461	246	577	5	582	396
topo3	1755	3257	291	575	14	589	920
topo4	1239	3257	530	1408	9	1417	1426
topo5	3257	6461	333	768	4	772	506
topo6	1239	1755	453	1235	6	1241	1240
topo7	1239	6461	495	1428	8	1436	682

Table 5.1: Properties of the combined rocketfuel topologies

Teixeira et al. [TMSV03] have shown for the Sprint network that the diversity in the topology inferred with rocketfuel is higher than in the real topology. Thus, it is expected that the topologies used in our simulations are favorable to the computation of pairs of disjoint paths compared to real ISP topologies. Consequently, if pairs of disjoint paths cannot be found in our simulations by the distributed path computation technique, the same technique is not likely to find such diverse paths in real ISP networks.

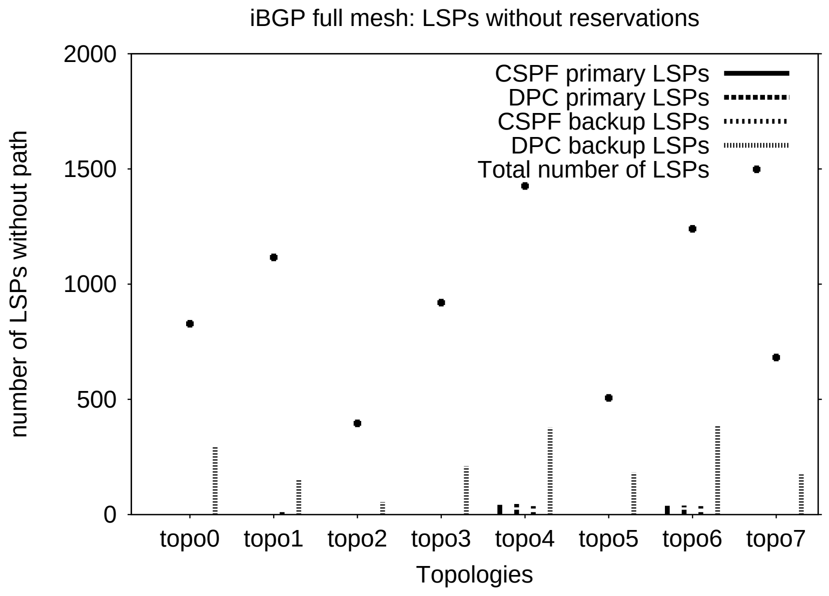
With the techniques described in section 5.2, we compute primary and backup paths with a 100ms delay constraint, suitable for Voice over IP (VoIP) traffic. We compute LSPs with and without 100Mbps bandwidth reservations. That is, for each primary path, we compute an end-to-end link and node disjoint path with the same constraints as for the primary path, for protection purposes. The ability to create backup paths is used as an indication of the diversity of the paths available to the centralized and the distributed techniques.

Figures 5.5 and 5.6 show the number of LSPs that could not be established for each topology and each path computation technique. For each topology, the total number of LSPs to be established is indicated by a point. The first and third bars show the number of protected and, respectively, backup LSPs that could not be established with the CSPF algorithm. The second and fourth bars represent the same values for the DPC technique.

The top part of figure 5.5 presents the number of LSPs that could not be established for the simulations with a full mesh of iBGP sessions in the ASs and no bandwidth reservations associated with the LSPs. We note that all the protected and backup CSPF LSPs could be established for most of the topologies. Thus, a more elaborate disjoint path computation algorithm than CSPF is not necessary in this case. CSPF is a good approximation of a k-SPF algorithm [Gro04]. However, DPC could not always find a feasible path for the backup LSPs. The results of the same simulations but with one RR per POP, instead of a full mesh of iBGP sessions, are provided at the bottom of figure 5.5. Here, we observe that paths could not be found for most backup LSPs with the DPC technique. This illustrates the fact that RRs hide part of the BGP routes to their clients.

Figure 5.6 concerns simulations of the establishment of LSPs with bandwidth reservations. A full iBGP mesh is used for the figure on the top. There are RRs in the ASs for the simulations of the bottom figure. We note that some protected LSPs cannot be established with the CSPF algorithm due to the limitation on the link capacities in the topologies and the structure of the rocketfuel topologies themselves. The same observation applies to the LSPs computed with DPC. In addition, we observe that less protected LSPs can be established with the DPC technique relying on BGP routes than with the centralized CSPF computation.

These figures confirm that RRs have a large impact on the possibility to find alternative paths. The difference between the number of backup LSPs that could not be established with CSPF and DPC lies in the limited number of routes available with BGP at the nodes participating in the computation. We performed the same simulations with different orderings of the LSPs and observed the same behavior. Moreover, we did not observe a big difference in the number of established LSPs when removing the full mesh of iBGP sessions inside the POPs when using RRs. The difference mostly lies in the presence of the RRs inside POPs not in the way iBGP sessions are established in the POP. This is due to the fact that in our



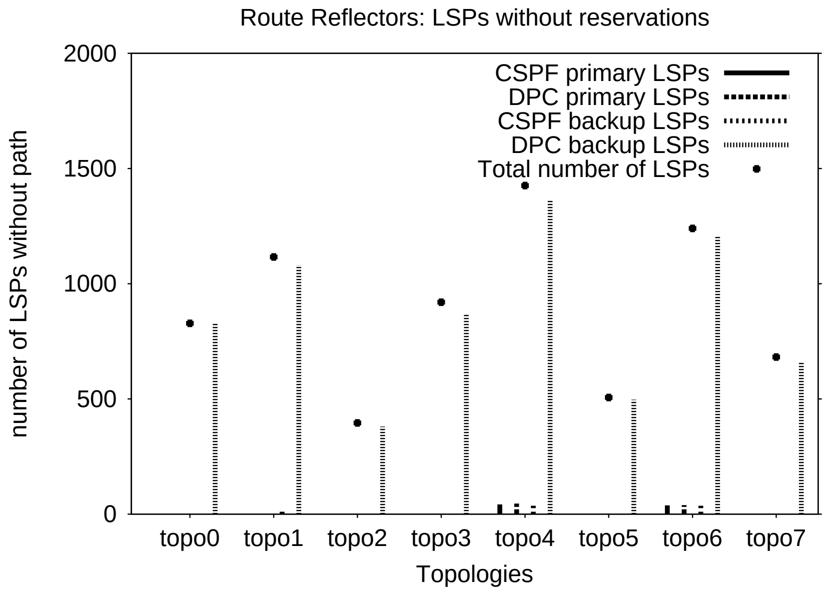
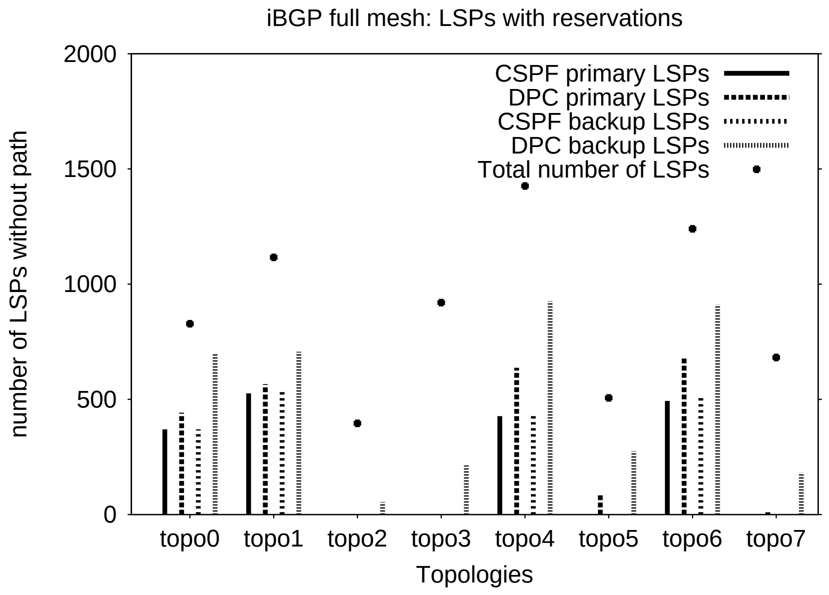


Figure 5.5: Number of paths that could not be established (without bandwidth reservations)

simulations there is at most one eBGP session, with the neighboring AS, per POP.

Figures 5.7 and 5.8 show the distribution of the difference in delay between CSPF and DPC LSPs. One curve compares the delay of the protected paths and the other curve compares the delay of the backup paths. Positive values indicate



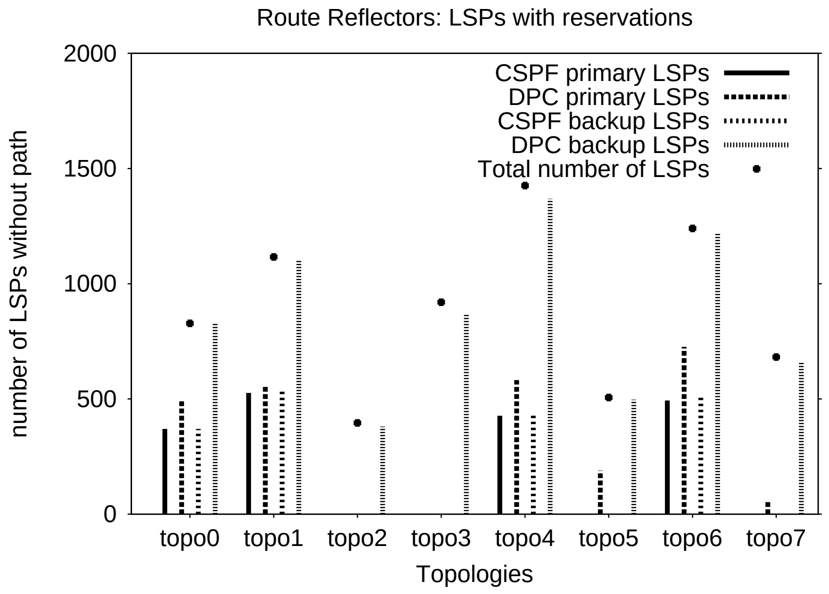


Figure 5.6: Number of paths that could not be established (with bandwidth reservations)

that the CSPF path has a shorter delay than the respective DPC path. Negative values occur when the DPC path has a shorter delay than the CSPF path between the same source and destination. These figures only shows the LSPs for which both the CSPF and the DPC paths could be computed. The results of figure 5.7 (figure

5.8) concern the establishment of LSPs without (with) reservations, respectively, on topology “topo4” with RRs inside the ASs.

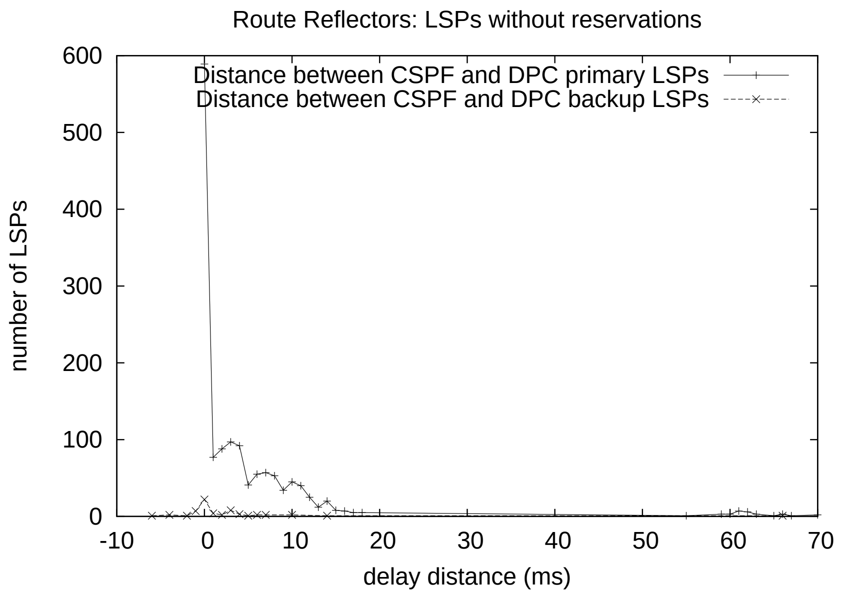


Figure 5.7: Delay distance (topo4)

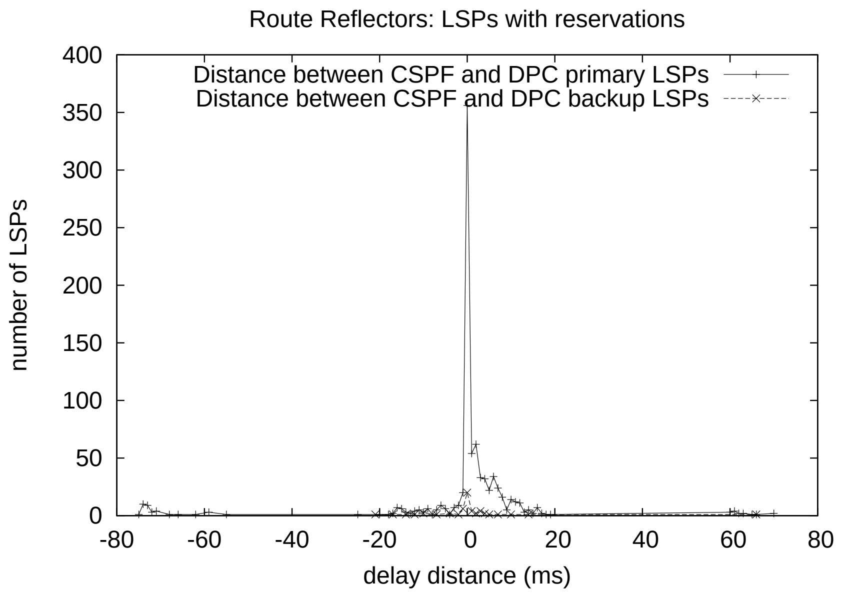


Figure 5.8: Delay distance (topo4)

First, we observe that there are a large number of LSPs with the same delay for the primary CSPF and DPC paths. This indicates that most of the paths have the same quality independently from the path computation technique. Most paths computed based on the information available with BGP (DPC technique) have a delay comparable to the paths obtained with CSPF. *Even though the path found by DPC is often of a similar quality than the CSPF path, for large topologies, the former is never found on the first try, i.e. crankback is used for every path computed by DPC.*

In figure 5.7, we see that some CSPF backup paths have a higher delay than their respective DPC paths (negative values). This behavior results from the lack of information available on the quality of the BGP routes and the local search of the DPC technique. The DPC algorithm chooses the NH reachable with the smallest delay. This is a local choice that may not be appropriate to minimize the end-to-end delay. For the backup path, the NHs used on the protected path are pruned from the topology. Bad NH choices, in terms of delay, made for the primary path, leave better alternatives for the backup path. Thus, the backup path may eventually follow the same path as the primary CSPF path.

In figure 5.8, we note that some DPC primary paths may have a shorter delay than the respective CSPF primary path when LSPs with bandwidth reservations are established. This results from the different distribution of the paths on the topologies with both computation techniques. With CSPF, the links with low delay will be used first. When there is no bandwidth left on these links, links with higher delay will be used resulting in a degradation of the end-to-end delay of the paths. Since DPC may perform bad choices based on local search, links with low delay may not be used by the first LSPs to be established. This leaves paths with low delays for subsequent LSPs.

We have seen in this section that nodes along LSPs are not able to contribute to the successful establishment of many backup LSPs, especially when RRs are present. Now, we look at the distributed computation of the paths by a different entity than the PE routers and the ASBRs as proposed in section 4.2.3. We take the same topologies as above, with one RR per POP, and the same BGP configurations. We assume that the RRs are able to compute the inter-AS paths on behalf of their clients. Again, we compute paths for LSPs without bandwidth reservations and with reservations. The number of LSPs that could not be established, with each technique, are shown in figures 5.9 and 5.10. In figure 5.9, we have the results for LSPs without bandwidth guarantees. In figure 5.10, we find the results for the simulations with bandwidth reservations. We observe that the results for the centralized CSPF computation do not change from the results in 5.5 and 5.6. It is still the same computation. However, when we compare the number of LSPs that cannot be established with DPC performed by RRs to this number with DPC performed on the path of the LSP (figures 5.5 (bottom) and 5.6 (bottom)), we observe less establishment failures with RRs performing the path computation. This is true for the establishment of LSPs without bandwidth reservations (figures 5.5 (bottom) and 5.9) as well as for the establishment of LSPs with bandwidth reservations (figures 5.6 (bottom) and 5.10). In our simulations, the RRs possess more routes than their clients because a BGP router does not have multiple eBGP sessions. In addition, there is a single eBGP session per POP. Thus, the RR-clients learn at most one route through eBGP or the intra-POP full-mesh and the other route from their RR. The RR, however, may learn one route from each POP as well as one route from one of its clients, inside its POP. We note that even if there are less establishment failures with RRs acting as PCEs, there is still room for improvement. The number of established disjoint LSPs that are computed distributedly is still below the number of disjoint LSPs that can be established based on the centralized CSPF computation.

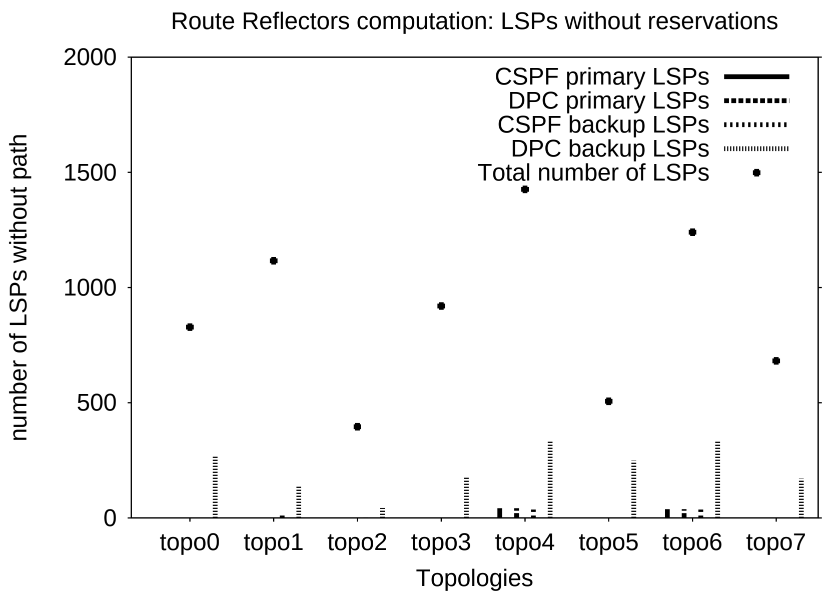


Figure 5.9: RR computation: Number of paths that could not be established (without bandwidth reservations)

5.4 Suggestions on TED Content

In section 5.3, we have seen, by means of simulations, that the BGP routes available at the nodes participating in the distributed computation have an impact on the constrained LSPs that can be established. First, we showed that the configuration of the iBGP sessions influences the diversity of the routes available at the different nodes inside the domain. We observed that a full-mesh of iBGP sessions was better than the use of RRs for path diversity. Secondly, we noted that in a configuration with RRs all nodes are not equal with regard to constrained path computation. In our simulations, the RRs possess more BGP routes than their clients. The study on the diversity of the BGP routes inside a real Tier-1 network, performed by Uhlig

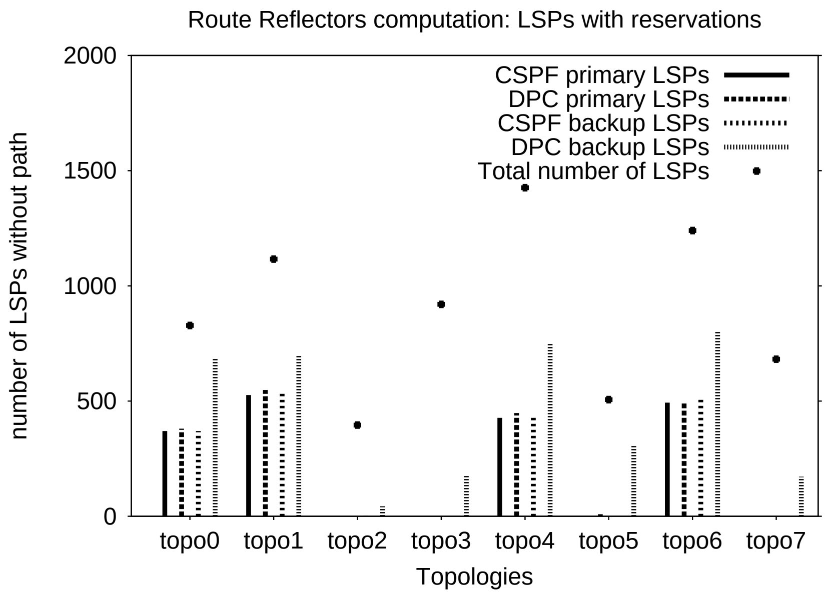


Figure 5.10: RR computation: Number of paths that could not be established (with bandwidth reservations)

et al. [UT06], confirms our observation. They noted that only top-level RRs have a very high diversity while most routers know only a single route for a prefix. Therefore, we deduce that RRs are more suitable for successful path computation than regular nodes that are on the path of the LSP.

In appendix B, we illustrate with an example that the use of the BGP next-hop self option may also, in case of multiple eBGP sessions on the same router, have an impact on the LSPs that can be computed by a distributed technique. If the egress ASBRs or the routes available at these nodes are not used by the distributed path computation, less interdomain paths can be used for the establishment of LSPs.

In order to avoid these BGP configuration dependence problems, we suggest the use of a node responsible of constrained path computation for the other nodes in the AS, the PCE. In addition, we recommend to collect all the BGP routes learned inside an AS at the PCE[4]. The result of the distributed path computation performed by PCEs, instead of the nodes belonging to the path of the LSP, with such TED content becomes independent of the topology of iBGP sessions in the AS and of the configuration of the BGP next-hop self option (see appendix B). Moreover, all the topology information gathered inside an AS through BGP is used by its PCE. Thus, better results than with PCEs collocated at RRs may probably be achieved.

In the remaining of this thesis, we apply these recommendations. We do not

[4]In this thesis, we assume that there is a single PCE inside each AS with the domain of the PCE covering the AS. There may be multiple PCEs covering the same domain. In this case, all these PCEs learn all the BGP routes learned inside their domain.

rely on nodes belonging to the LSP's path for the path computation. We assume the existence of a PCE that participates in distributed path computation on behalf of the nodes inside its domain. As suggested this PCE possesses all the BGP routes that are learned inside the AS. We apply these recommendations for both ERO expansion and cooperative PCEs. If the trapping problem does not occur often in computer networks as asserted in [Gro04], the number of LSPs established with cooperative PCEs should be similar to the number of LSPs established with ERO expansion based on the same TED content[5]. Thus, cooperative PCEs also benefit from these recommendations.

5.5 Conclusion

We evaluated in this chapter the importance of the TED content on the establishment of interdomain LSPs. We showed that BGP-related limitations make the problem of computing constrained end-to-end LSPs difficult, namely the topological information hiding and the unawareness of end-to-end metrics by BGP when choosing its best route.

We illustrated our case by comparing two different LSP computation techniques. The first technique, a centralized one, is based on CSPF. It assumes that the intradomain topology of the ASs crossed by the LSP is known by the central entity performing the computation, the global PCE. The second technique is fully decentralized. It relies on the BGP routes present locally in the routers as well as on the topology of the local domain.

In large topologies, our simulations showed that the establishment of the constrained LSPs with the DPC technique using nodes on the path of the LSP always required to crankback due to the lack of QoS information available through BGP. Thus, designing BGP-based interdomain LSPs computation techniques with guarantees will always face the fundamental trade-off between the scalability of the interdomain path computation and the quality of the paths found in terms of the considered metrics. This trade-off will be studied in subsequent chapters.

Our simulations show that the decentralized technique is not able to provide end-to-end link and node disjoint paths only based on the BGP routes available locally. In addition, they have highlighted the fact that certain nodes, the RRs, are more capable of computing diverse paths than their clients.

These simulations showed the impact of BGP configurations on the diversity of the paths that can be computed distributedly. From these observations, we made recommendations on the content of the TED inside PCEs. The objective of these recommendations is for the PCE to be able to compute paths independently of the iBGP topology of its domain. Additionally, they enable the PCE to gather as much topology information as available with BGP inside the domain and, thus,

[5]Cooperative PCEs as proposed in [VZB06] perform better than ERO expansion in case of a trap topology

to have the key of computing the best paths that are possible with BGP. These recommendations are used in the remaining of this document.

Chapter 6

Delay Constrained Interdomain LSPs

The QoS properties of the paths, such as the delay and bandwidth, to reach a destination outside an AS are currently not known inside the AS. This information is not distributed by BGP, the current interdomain routing protocol. Extensions to BGP have been proposed in order to distribute such information [Bou05]. In this thesis, we do not rely on such extensions. We believe that distributing QoS with the BGP routes will increase the amount of update messages exchanged, the size of these update messages and the size of the BGP routing tables. The size of the BGP routing tables is already a problem. [BGT02] say that the BGP routing table sizes have grown by a factor of two between 1997 and 2002. Moreover, if QoS is used in the selection of the BGP routes, this will most likely create instabilities of the interdomain routes and reduce the diversity of the routes that are received for a prefix. If the quality of the routes is determined from the same criterion at all routers, only a very few good routes will be distributed for a prefix in the Internet. Last, deploying QoS extensions to BGP is not an easy task. For the QoS information to be relevant, all the BGP routers that forward an update message should support the QoS extensions. A strong incentive has to be found for SPs to update their BGP software. The update messages without QoS information related to the complete path will most likely not be used for the establishment of interdomain LSPs or heuristics have to be found to infer the QoS of routes based on incomplete information. In this thesis, we rely on heuristics to estimate the QoS properties of a BGP route. We do not assume changes to the BGP routing protocol. Our heuristics can also be used with BGP QoS solutions to palliate with incomplete or inaccurate QoS information. In this chapter, we propose that a PCE performing ERO expansion relies on heuristics in order to choose an appropriate BGP NH among the NHs announced for the destination. In addition, we propose that a cooperative PCE relies on heuristics to assign preferences to the downstream domains and thus contact PCEs inside the domains with a higher preference.

It is important that PCEs use good heuristics. If a bad choice is performed by the heuristic at some PCE, a downstream PCE may not be able to complete the computation of the path. In this chapter, we propose two heuristics that can be used by the PCEs during the computation of LSPs. The heuristics try to determine the NHs that are along short delay paths because the LSPs considered in this chapter are subject to maximum end-to-end delay constraints.

6.1 Heuristics for BGP NH Ranking

In this section, we describe the heuristics that we propose to use in order to assign preferences to the NHs available from the BGP routes.

6.1.1 Nearest NH

We call our first NH ranking heuristic “nearest NH”. Two link metrics are provided with ISIS-TE/OSPF-TE : the classical IGP metric and a TE metric. The IGP metric is usually set to the link bandwidth. We propose to set the TE metric of each link to its propagation delay. The PCE ranks the NHs available for the destination based on the delay of the shortest path, from the ASBR to the NH, with enough bandwidth to support the LSP. The TE metric is used for the computation of the shortest path.

6.1.2 Vivaldi

[CK06] describes two categories of techniques to estimate the latency between two nodes. The issue is to be able to determine the delay between any two nodes without having to measure the delay between all pairs of nodes. In proxy-based methods, this delay estimation relies on measurements between the source and a proxy, this proxy and another proxy, the latter proxy and the destination. The embedding-based methods rely on the coordinates of the nodes in an Euclidean space. The delay between any two nodes is estimated by the Euclidean distance between these two nodes. The coordinates of the nodes are computed based on a few latency measurements with given nodes called landmarks.

Preferring the “nearest NHs” in terms of the delay, as in the first heuristic, does not ensure that the end-to-end delay of the paths through the preferred NHs will be low. The path segment downstream of a NH with a high ranking may have a long delay, as we will see in figure 6.1. Thus, the heuristic proposed in this section relies on a delay estimation of the paths through the candidate NHs up to the tail-end of the LSP.

We use a virtual coordinate system, called Vivaldi [DCKM04], to estimate the delay of a path between two nodes. In this coordinate system each node computes its coordinates based on RTT measurements with a limited number of other nodes. Nodes connected with a low delay path will have neighboring coordinates while nodes connected through a higher delay path will be further apart.

In the heuristic presented in this section, we prefer the NHs that are along the path with the smallest delay estimation toward the tail-end D, to minimize the delay of the remaining portion of the path to D. Thus, for an ingress ASBR I_c inside an AS AS_c, we prefer the ingress ASBR I_d inside a downstream AS AS_d such that

$$delay(I_c, I_d) + distance(I_d, D) = \min_{I_j \in NH} (delay(I_c, I_j) + distance(I_j, D))$$

where NH is the set of potential NHs for tail-end D, $delay()$ is the delay of the ISIS/OSPF path computed with the TE metric and $distance()$ is the distance between two points in the virtual coordinate space.

In our simulations, each node computes its coordinates in a two-dimensional Euclidean space augmented with an height, noted $2d + h$, as proposed in [DCKM04]. The distance between two nodes with coordinates (x_1, y_1, h_1) and (x_2, y_2, h_2) in the $2d + h$ space is the sum of the distance of the first node to the plane (its height, h_1), the Euclidean distance between the coordinates of the two nodes in the plane ($\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$) and the distance from the plane to the second node (the height of the second node, h_2).

In order to compute the preference of the candidate NHs, the PCE needs to know the coordinates of each NH and of the LSP's tail-end. For this purpose, we assume that after the computation of its coordinates, each node stores these coordinates inside its Domain Name Server (DNS), as proposed in [dUB05b]. The PCE requests the coordinates of the candidate NHs and the destination from the DNS.

6.2 Use of Heuristics by the Path Computation Techniques

The heuristics presented in the previous section assign preferences to the NHs that are available for a given destination. In this section, we show how we propose to use these heuristics in the computation of interdomain constrained paths.

6.2.1 ERO Expansion

The NHs available for the destination of an LSP are assigned preferences based on the heuristics defined in section 6.1. In ERO expansion, a path segment is computed to the preferred NH, first. If a path segment to the preferred NH that respects the constraints for the LSP is found, the LSP is established to this NH. If the preferred NH does not lead to a feasible path, with respect to the constraints, the second best NH is tried, and so on. In ERO expansion, one NH at a time, the best NH according to an heuristic, is selected for an attempt of LSP establishment among the set of NHs that have not been tried.

In figure 6.1, we illustrate the selection of the NH by the two heuristics for an LSP entering $AS2$ at router $R2$ with tail-end $R8$. There are two candidate NHs, $R5$ and $R6$, for destination $R8$. The PCE inside $AS2$ prefers $R5$ over $R6$ with the “nearest NH” heuristic because the shortest delay path from $R2$ to $R5$ is 2 and the shortest delay path from $R2$ to $R6$ is 7. With the “vivaldi” heuristic, the PCE prefers $R6$ instead of $R5$ because the delay estimation[1] of the path from $R2$ to $R8$ transiting through $R6$ is $7 + \sqrt{(37-34)^2 + (18-10)^2} = 15.5$ and the delay estimation of the path transiting through $R5$ is $2 + \sqrt{(61-34)^2 + (78-10)^2} = 75$. The path from $R1$ to $R8$ obtained with the "nearest NH" heuristic is $R1 - R2 - R4 - R5 - R7 - R6 - R8$ with delay of 44 ms. On the other hand, the path $R1 - R2 - R4 - R3 - R6 - R8$, resulting from the computation with the "vivaldi" heuristic has a shorter delay of 9 ms.

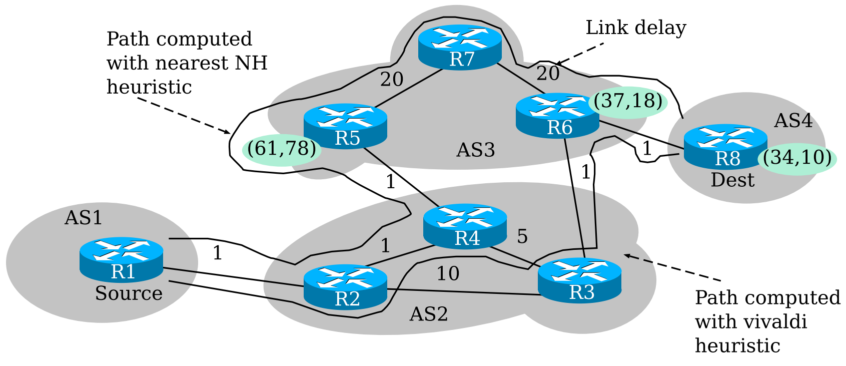


Figure 6.1: ERO expansion: nearest NH versus vivaldi

6.2.2 Cooperative PCEs

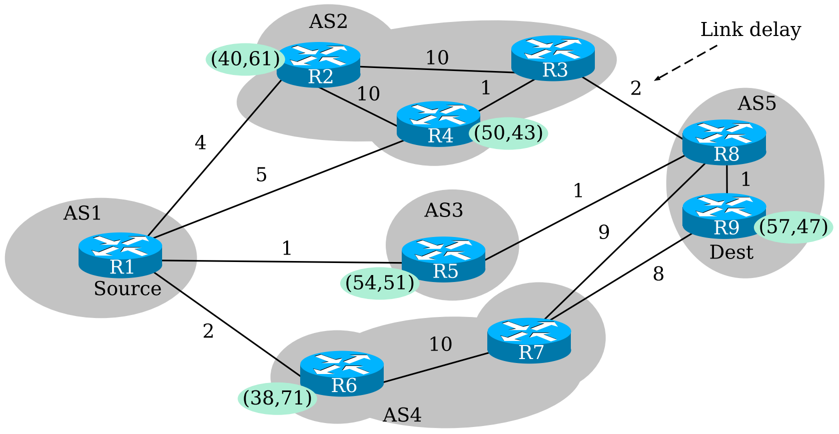
A cooperative PCE could use a heuristic to rank the NHs. Then, based on this classification, it determines n different downstream domains that contain the preferred NHs. The PCE only contacts PCEs in the domains of these best NHs. This enables to reduce the number of AS paths explored in the computation of a path for an LSP, compared to an exploration of all the downstream paths. As a consequence, the number of messages exchanged between PCCs and PCEs is also reduced.

It is important to use a good heuristic in order to find a path that respects the constraints of the LSP. Moreover, since the objective of the computation by means of cooperative PCEs is to find the shortest path, the heuristic should contribute to this objective. This should be enforced by the heuristics considered in this chapter because they aim at determining the NHs along the shortest delay path.

In figure 6.2 we illustrate the use of the nearest NH and vivaldi heuristics in the path computation by means of cooperative PCEs. An LSP with minimum delay has to be established between $R1$ and $R9$. There are 4 NHs available for destination $R9$. These NHs are $R2$, $R4$, $R5$ and $R6$. Based on the delay of the link from $R1$ to each NH, the nearest NH heuristic classifies the NHs as follows. First, there is $R5$ with the highest preference, second $R6$, third $R2$ and finally $R4$ with the lowest

[1]In this example, we consider $2d$ coordinates. The delay estimation between two nodes in this $2d$ space is the Euclidean distance between the coordinates of the two nodes.

preference. The vivaldi heuristic assigns different preferences to these NHs. These preferences rely on a delay estimation of the paths between R1 and the destination via each NH. NHs on paths with low delay estimations are preferred over NHs on paths with a longer delay estimation. With the vivaldi heuristic, the preferred NHs are, by order of preference, $R5$, $R4$, $R2$ and $R6$.



Delay	Nearest NH pref.	Delay estimation R1-R9 via	Vivaldi pref.
R1-R2: 4	R2: 3	R2: 4 + 22.02 = 26.02	R2: 3
R1-R4: 5	R4: 4	R4: 5 + 8.06 = 13.06	R4: 2
R1-R5: 1	R5: 1	R5: 1 + 5 = 6	R5: 1
R1-R6: 2	R6: 2	R6: 2 + 30.61 = 32.61	R6: 4

Figure 6.2: Cooperative PCEs: nearest NH versus vivaldi

Let us assume that each cooperative PCE only contacts a single downstream PCE. Then, the PCE of $AS1$, in figure 6.2, sends a PCReq message to the PCE in $AS3$ independently of the heuristic, because $AS3$ contains the NH that is preferred by both heuristics. If PCEs are contacted in two downstream domains, then PCEs in $AS3$ and $AS4$ are contacted by the PCE in $AS1$ if it relies on nearest NH for NH ranking. Otherwise, with the vivaldi heuristic, PCReq messages are sent to PCEs in $AS3$ and $AS2$. Because two of the NHs belong to $AS2$. These two NHs are considered in the computation at the PCE in $AS2$. Finally, if the PCE in $AS1$ sends PCReq messages to PCEs in three downstream domains, all the NHs available for $R9$ in $AS1$ are considered in the path computation.

6.3 Evaluation of the Heuristics

In this section, we evaluate the four interdomain path computation techniques presented in chapter 4 in the computation of paths with a maximum end-to-end delay constraint. The objective is to determine the quality of the paths computed by the different techniques, in terms of delay. We interest ourselves at the amount of traffic that can be carried inside the topology with each path computation technique. In addition, we aim at evaluating the amount of signalling required by the techniques.

First, we compare the use of the nearest NH and vivaldi heuristics to select the NH in the ERO expansion technique. We determine the amount of crankback that is required to compute path with ERO expansion. Moreover, we position the ERO expansion technique with regard to the techniques used as benchmark, that is, global PCE path computation and IP forwarding.

Then, we look at the performance of the cooperative PCEs when a PCE contacts the PCEs in all the downstream domains for the LSP's tail-end. This set of downstream domains is determined from the BGP routes in the TED of the PCE. We determine the quality achieved for the computed paths, in terms of delay. Moreover, we provide boundaries on the number of signalling messages between PCCs and PCEs.

We estimate the loss in path quality when using the vivaldi and nearest NH heuristics in order for a cooperative PCE to contact a limited number of downstream PCEs. We assess the reduction in signalling that results from such practice.

Finally, we compare the quality of the paths and the signalling required by the ERO expansion and the cooperative PCE techniques.

In this section, we present the results of simulations on two types of topologies[2]. First, we use topologies composed of 5 transit ASs to evaluate our heuristics in a small environment with MPLS deployed between and inside all the ASs. Such an environment is conceivable today. Then, we apply the path computation techniques on a larger topology composed of 20 transit ASs, as in the core of the Internet [SARK02], to evaluate the techniques in a large scale deployment of inter-AS MPLS LSPs.

6.3.1 Simulation's Settings

In our simulations, we consider that a PCE receives all the BGP routes learned inside the AS, as suggested in chapter 5. These routes populate its Traffic Engineering Database. They are used for the computation of interdomain constrained paths.

In addition, the ingress ASBRs of neighboring ASs and the inter-AS links are known inside the local AS. The NHs of the BGP routes belong to the downstream domains. They are ingress ASBRs. That is, we assume that the BGP next-hop self option is not used because of the cooperative PCEs computation technique. In [VZB06], cooperative PCEs have to be able to compute paths segments ending in the downstream domains, in order to be able to assemble them with the path segments received from the downstream PCEs.

The topologies used for the simulations are generated with the transit-stub model of the GT-ITM tool [ZCD97]. First we generated 5 topologies each composed of 5 transit ASs. In these topologies, each transit AS is composed of approximatively 50 routers. The links inside the transit ASs are generated randomly with

[2]The topologies and scripts used to provide the results presented in this section are available to the research community at the following URL: `http://totem.info.ucl.ac.be/tools`.

the parameters suggested by the authors of GT-ITM in [ZCD97]. GT-ITM attaches one stub AS to each router in a transit AS and randomly adds 250 extra links between the transit and the stub nodes. Each stub AS only contains one router. This router is the end-point of the LSPs established on the topology.

We group the stubs in classes that contain all the stubs attached to the same providers. We only keep one stub from each class to reduce the simulation time. It results in topologies with an average of 27 stubs. The nodes in these selected stubs and the nodes inside the transit ASs are placed by GT-ITM in an Euclidean plane. This placement is used to set the delay of the links. In our topology, the delay of a link is directly proportional to the Euclidean distance between its two end-points. In addition, we assign the same bandwidth to all the links.

In our simulations, we establish a full-mesh of LSPs between the routers in the stub ASs. Such a full-mesh could correspond to a very large interdomain BGP/MPLS VPN service. We establish the LSPs in one direction only. All LSPs are subject to the same maximum end-to-end delay constraint (1900 units [3]).

The delay constraint is determined as follows. For each LSP to be established, we computed the shortest path in terms of delay from the head-end to the tail-end node, on the complete topology and without BGP policies and filtering. We set the delay constraint of all the LSPs to the same value. This value is a round value just above the maximum delay of the resulting paths to ensure that, for each LSP, a path respecting the delay constraint exists in the topology.

We use the C-BGP simulator [QU05] to compute the BGP routing tables of the nodes. The routers inside stub ASs are configured not to advertise routes received from other ASs. Thus, stub ASs do not provide transit service. Transit ASs do not filter out the routes advertised to neighboring ASs. This ensures that each AS receives at least one route for each destination.

The second topology is composed of 20 transit ASs as the core of the Internet. It is generated by the method described earlier for the topologies with 5 transit ASs. Again, the transit ASs are composed of 50 nodes and all links have the same capacity. This topology has 411 stub ASs. We try to establish 84255 LSPs on this topology. Again all LSPs are subject to the same end-to-end delay constraint (3300 units). This constraint is determined by the procedure described for the small topologies.

We have said in section 6.1.2 that the vivaldi coordinates rely on RTT measures. In our simulations, we rely on RTT measures between the LSPs' end nodes, between LSPs' end nodes and the ASBRs inside the same ASs, between LSPs' end nodes and the ASBRs inside the neighboring ASs, and between all pairs of ASBRs.

6.3.2 ERO Expansion

In this section, we compare the use of nearest NH and vivaldi heuristics by the ERO expansion path establishment and computation technique. We interest ourselves to

[3]The units used for the delay constraint and the delay of the links depends on the scale used to generate the topology.

the number of LSPs established, the end-to-end delay of the computed paths and the amount a crankback that occurs in the computation of the paths. We first look at the topologies with 5 transit ASs. Then, we consider the topologies composed of 20 transit ASs.

The number of LSPs that can be established by a global PCE, ERO expansion and the two heuristics, as well as the number of IP forwarding paths below the end-to-end delay constraint, on the topologies with 5 transit ASs, are shown in figure 6.3[4]. We make the same observations for each of the five topologies. First, the total number of LSPs to be established on each topology is equal to the number of LSPs established by a global PCE performing a CSPF computation. The maximum end-to-end delay constraint of the LSPs has been set in order for this to be true, when no bandwidth reservation is associated with the LSPs. Then, the number of IP forwarding paths with a delay respecting the constraint of the LSPs is below the number of LSPs that can be established with each of the other path computation techniques. Third, the number of LSPs established with ERO expansion is independent of the heuristic that is used. This comes from the fact that no bandwidth reservations are required for the LSPs. The state of the network is the same before the establishment of all the LSPs. Thus, because both heuristics rely on the same Traffic Engineering Database (TED) content, if a path that respects the delay constraint can be found by one heuristic it can also be found by the other heuristic, even if, both heuristics compute different paths for the preceding LSPs.

Because the simulations on the 5 different topologies containing 5 transits ASs provide similar results, we mostly consider only one of these topologies in the remaining of our analysis.

Figure 6.4 shows the cumulative distribution for the end-to-end delay of the LSPs with paths computed by the different techniques. We consider the end-to-end delay of the paths computed by a global PCE, the delay of the IP forwarding paths as well as the delay of the paths established with ERO expansion. Each curve represents the results for a different path computation technique. There is one curve for each heuristic used by ERO expansion. Each curve shows the number of LSPs (y-axis) whose path has a delay lower than or equal to a given delay (x-axis).

We observe from figure 6.4 that CSPF performed by a global PCE enables to find the shortest delay paths. There are more LSPs with a low delay path with the global PCE than with the other computation techniques. The global PCE computes the shortest delay paths based on the complete knowledge of the topology. This computation does not imply BGP route filtering and routing policies contrary to the other path computation techniques. In addition, we note that there are IP forwarding paths with a low delay. In fact, there are more LSPs with a delay below 1600 units[5] with IP forwarding than with both ERO expansion heuristics. The delay of the IP forwarding paths is not bad because BGP routers use the IGP cost to the NH in the selection of a route. Here, the IGP cost of a link is set to the delay of

[4]The bandwidth of the links is set to 10 Gbps in each topology used in this chapter.

[5]This unit depends on the scale used to generate the topology.

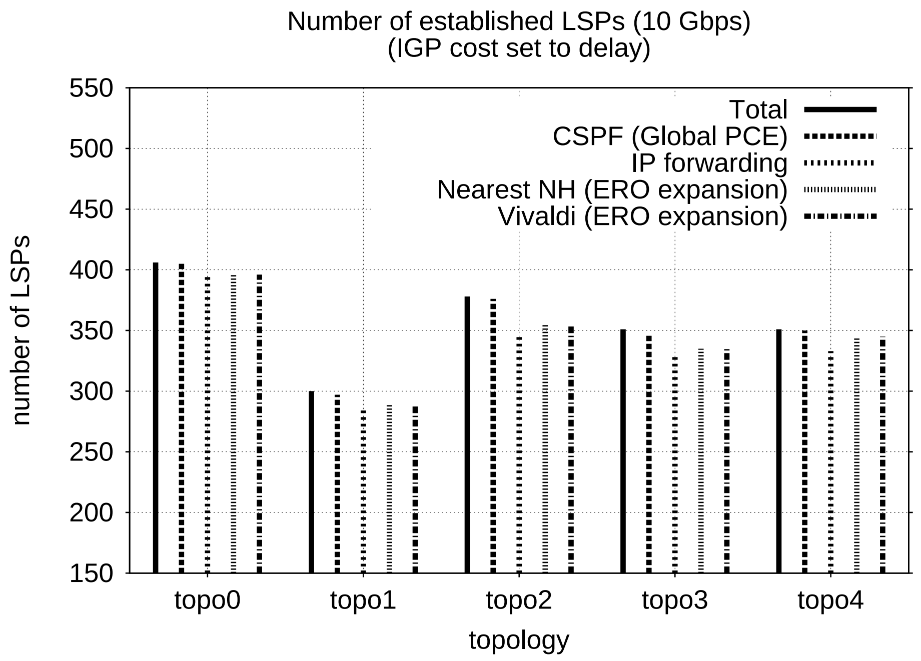


Figure 6.3: Number of LSPs established with ERO expansion on topologies with 5 transit ASs

the link. Thus, from two routes with the same preference with regard to the rules preceding the IGP cost rule, the route with smallest delay to the NH is preferred, in our simulations. This leads to short delay BGP routes. Finally, we note from figure 6.4 that the vivaldi heuristic is able to find more shorter delay paths than the nearest NH heuristic.

Now, we look at the amount of crankback performed by ERO expansion with our two heuristics. The ERO expansion technique relies on crankback to undo bad choices made by the NH selection heuristic. With ERO expansion, all the available inter-AS paths have to be tried in order to determine that no suitable path can be found to accommodate an LSP. However, once a path respecting the constraints is found, the LSP is established along this path. Therefore, we distinguish LSPs that are established on the topology from LSPs for which a suitable path cannot be found in our analysis of the crankback. The amount of crankback that occurs during the successful establishment of an LSP indicates the ability of the heuristic to determine appropriate NHs for given LSP constraints. Additionally, the amount of crankback is equal to the number of path segments along which unnecessary state is created for the LSP inside each node of the segment. The states created along these path segments is not necessary because these segments do not belong to the final path for the LSP. We note that these states are removed as soon as a

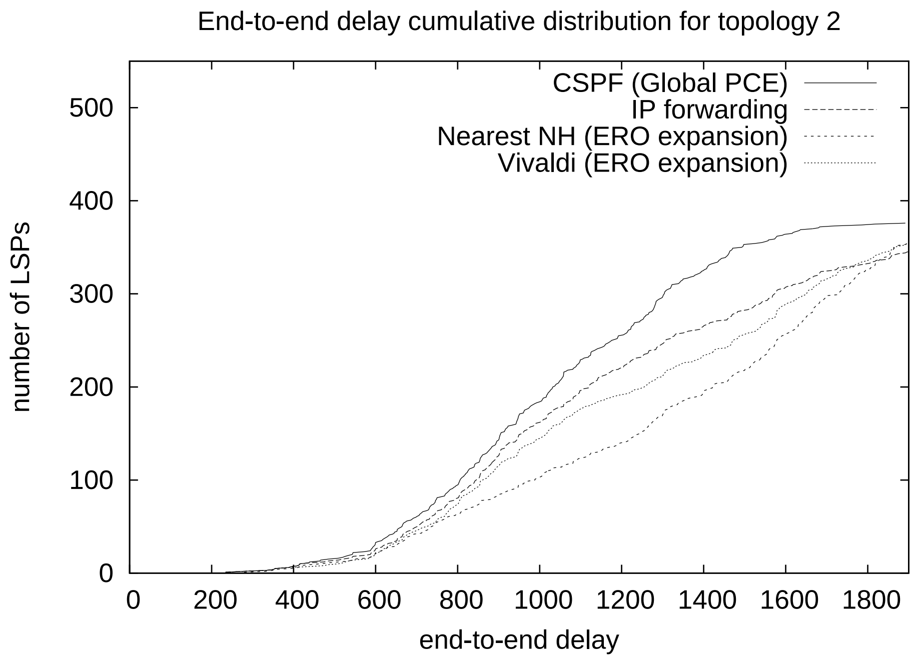


Figure 6.4: Path quality with ERO expansion on topology with 5 transit ASs

Path Error message is sent upstream along these segments, during the crankback process.

Figure 6.5 shows the cumulative distribution of the number of crankback that occurs during the computation of LSPs that can be established. We see that less crankback is required with the vivaldi heuristic than with nearest NH. The maximum number of crankback with vivaldi is 15. This number is 24 with the nearest NH heuristic. Moreover, there are 90% of the established LSPs that require less than 1 crankback, with vivaldi. With nearest NH, 80% of the established LSPs require less than 2 crankbacks and 90% require less than 6 crankbacks. Thus, the vivaldi heuristic is better that nearest NH to select a suitable NH for LSPs subject to end-to-end delay constraints.

The amount of crankback required in order to decide that a suitable path cannot be found for an LSP is illustrated in figure 6.6. There are 23 LSPs that cannot be established with ERO expansion on topology 2. These LSPs cannot be established due to BGP route distribution and filtering. As noted before, this number is the same for both heuristics. Here, the two heuristics give the same results. This comes from the fact that the establishment of an LSP is tried along the same path segments. Crankback occurs at the same nodes even if the nodes are tried in a different order. In figure 6.6, there are two LSPs that require 31 crankbacks before being able to determine that they cannot be established. This is the maximum

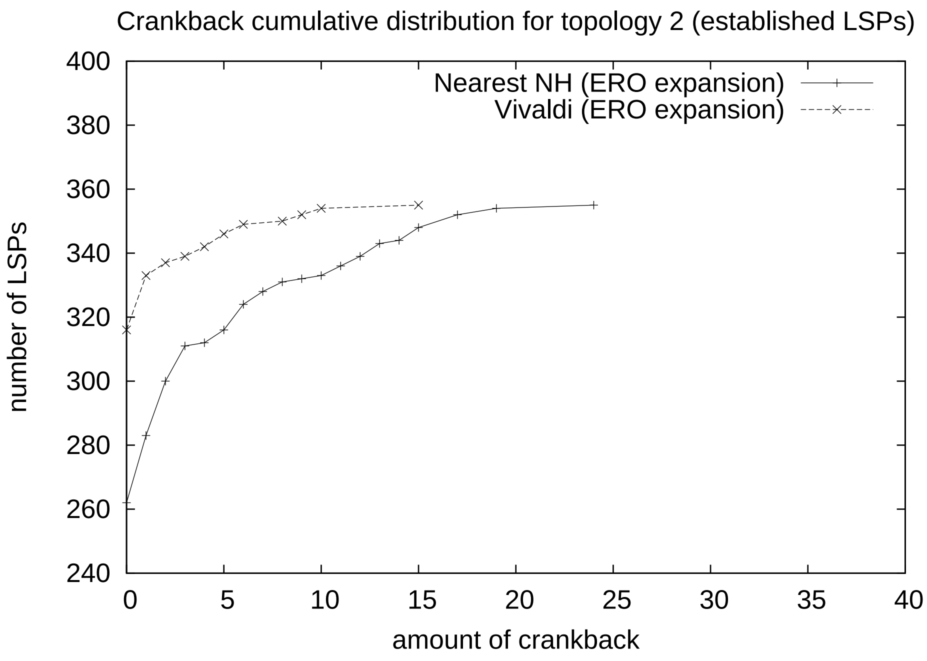


Figure 6.5: Amount of crankback for LSPs established on topology with 5 transit ASs

number of crankback that is observed in the simulations on the topology composed of 5 transit ASs (topology 2). It is twice the amount of crankback compared to the LSPs established with the vivaldi heuristic. It is more difficult to determine that an LSP cannot be established than to establish the LSP when a path respecting the constraints exists. ERO expansion stops once a suitable path is found or all the available paths have been tried.

The simulations on the topology with 20 transit ASs give similar results than with the topology composed of 5 transit ASs. We see in figure 6.7 that more paths with a low delay are found by the global PCE than the other path computation techniques that rely on BGP routes. A path respecting the delay constraint is found for all the LSPs with the global PCE. We also observe that there are IP forwarding paths with a low delay. There are more LSPs with a delay below 3000 units with IP forwarding that with both ERO expansion heuristics. However, there is an IP forwarding path respecting the delay constraint for only 84% of the LSPs. Finally, we note that both heuristics give similar results concerning the end-to-end delay of the computed paths. There are slightly more low delay paths with vivaldi than nearest NH. For example, there are 6% of additional LSPs with a delay below 2500 units with vivaldi compared to nearest NH. The difference is not substantial. “nearest NH” is the simplest heuristic. It selects the NH only based on local delay

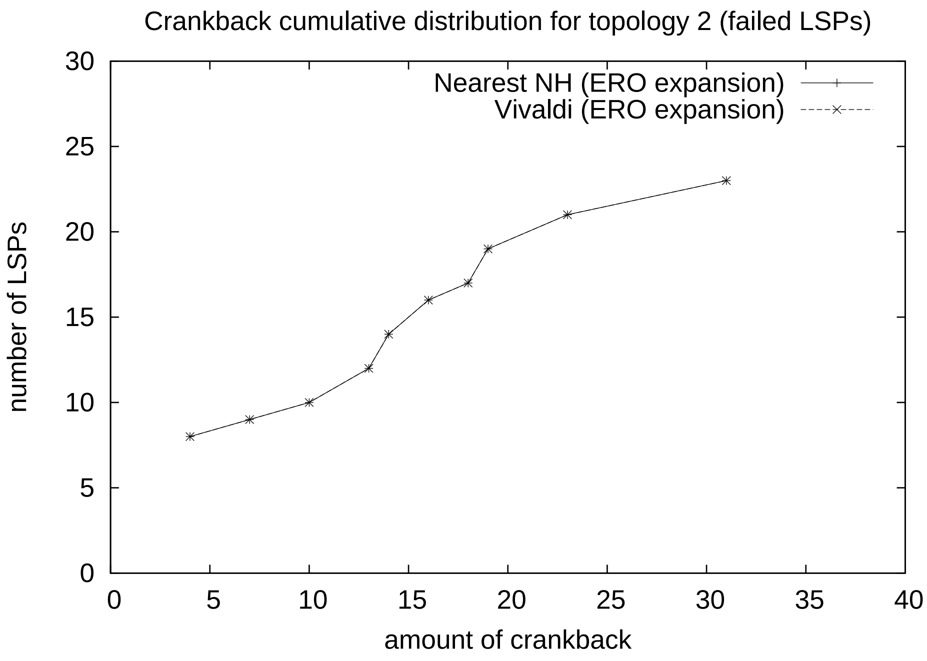


Figure 6.6: Amount of crankback for LSPs that cannot be established on topology with 5 transit ASs

information. The vivaldi heuristic requires to compute virtual coordinates. Then, NHs are selected based on local delay information and a delay estimation from the NH to the tail-end inferred from the vivaldi coordinates. Therefore, in terms of delay, it does not pay to use the vivaldi heuristic with regard to the simple “nearest NH” heuristic.

Concerning the amount of crankback for established LSPs, we observe in figure 6.8 that more LSPs require a small amount of crankback with vivaldi than with the nearest NH heuristic. No crankback is required for 60% of the LSPs with vivaldi. With nearest NH, 50% of the LSPs do not require crankback. The maximum amount of crankback is also higher with nearest NH than with vivaldi. It is equal to 373 with nearest NH and 293 with vivaldi. Moreover, only a small portion of the LSPs require a large amount of crankback. Only 10% of the LSPs established with the vivaldi heuristic require between 23 and 293 crankbacks. Between 44 and 373 crankbacks occur for 10% of the LSPs established with nearest NH. Finally, 80% of the established LSPs require less than 8 and 17 crankbacks for vivaldi and nearest NH, respectively.

Among the 84255 LSPs to be established on the topology with 20 transits ASs, there are 5111 LSPs that cannot be established with ERO expansion, independently of the heuristic. The amount of crankback required to determine that

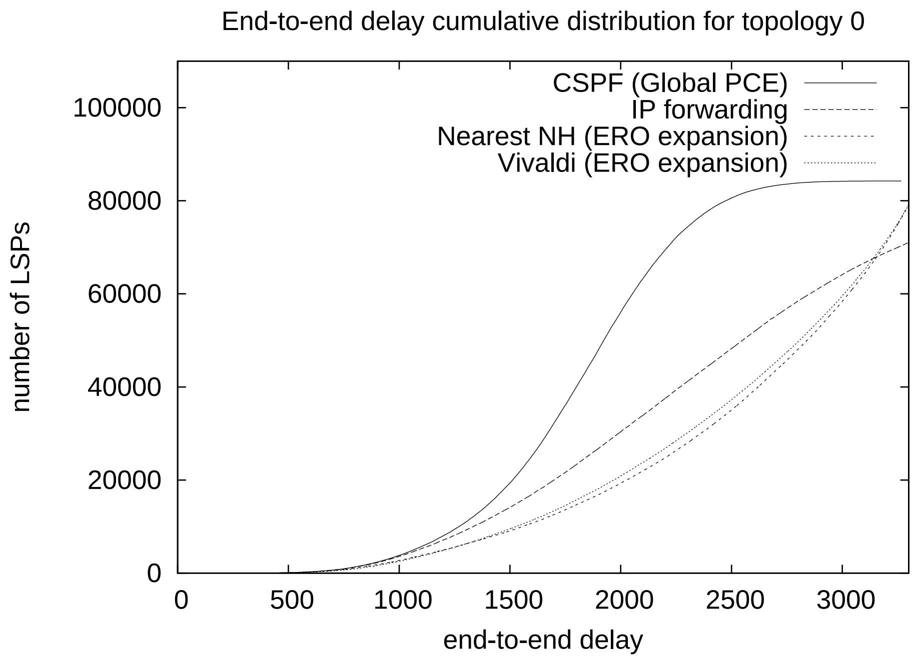


Figure 6.7: Path quality with ERO expansion on topology with 20 transit ASs

these LSPs cannot be established is illustrated in figure 6.9. The maximum number of crankback that we observe is 420. Between 158 and 420 crankbacks occur for 10% of the LSPs for which a path respecting the constraints cannot be found. Again we note that in order to determine that an LSP cannot be established, all the available inter-AS paths have to be tried while an LSP is established as soon as a suitable path is found. Thus, in general, more crankback is required in order to determine that an LSP cannot be established.

To conclude, the comparison of the two heuristics used by the ERO expansion technique has shown that the vivaldi heuristic is slightly better than the nearest NH heuristic when considering the end-to-end delay of the computed paths and the number of crankback. In addition, we have shown in this section that the end-to-end delay of the paths computed by a global PCE are lower than the delay of the paths computed with the other techniques that rely on BGP routes. This is due to BGP routing policies and the redistribution of the routes with BGP.

6.3.3 Cooperative PCEs

In this section we study the performance of the path computation technique that makes use of cooperative PCEs. We compare the computation with cooperative PCEs contacting all the available downstream PCEs to IP forwarding and CSPF in-

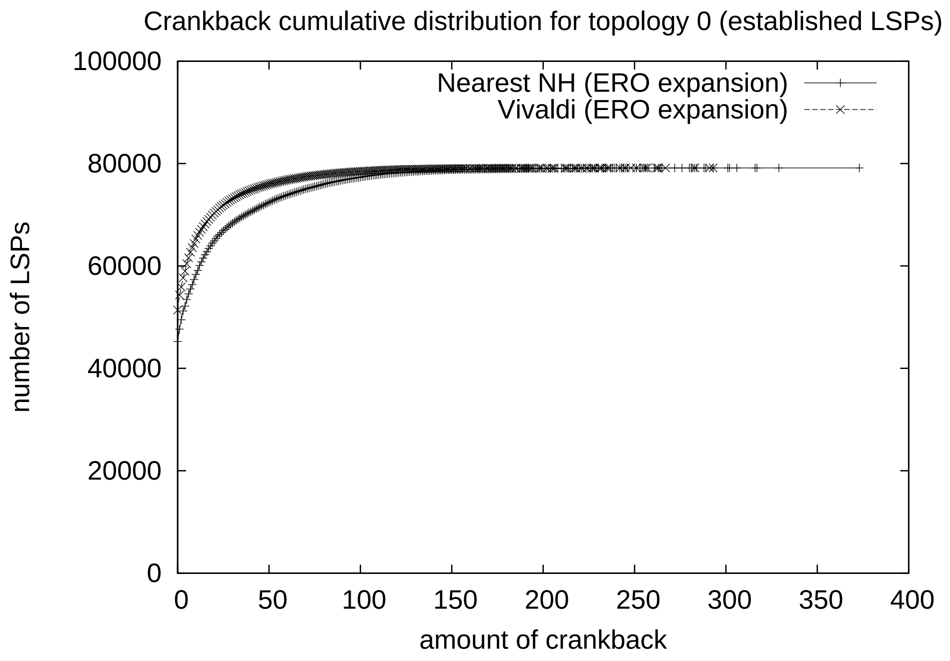


Figure 6.8: Amount of crankback for LSPs established on topology with 20 transit ASs

side a global PCE. Moreover, we compare the computation implying all the downstream PCEs of a participating PCE to the computation that uses the heuristics to select a set of PCEs to contact. We compare the results with one, two, three, . . . downstream PCEs that are contacted by each PCE. We compare the paths quality in terms of delay and the signalling overhead of the different path computation techniques and the different settings of cooperative PCEs.

First, we compare the end-to-end delay of the paths computed with cooperative PCEs contacting all the available downstream PCEs to the paths computed by the global PCE and the IP forwarding paths. The results obtained on a topology composed of 5 transit ASs are shown in figure 6.10. The curve presenting the results for the computation with cooperative PCEs is labeled "Coop PCE". We say that when each PCE contacts all its downstream PCEs, a complete exploration of the paths is performed. The "Coop PCE" curve presents the results for such a complete exploration.

We see in figure 6.10 that more paths with a low delay are computed by cooperative PCEs compared to the paths resulting from standard IP forwarding. In fact, the paths resulting from a complete exploration of the downstream PCEs are the shortest delay paths that can be computed based on BGP information. In addition, we spot that the global PCE is able to compute shorter delay paths than coopera-

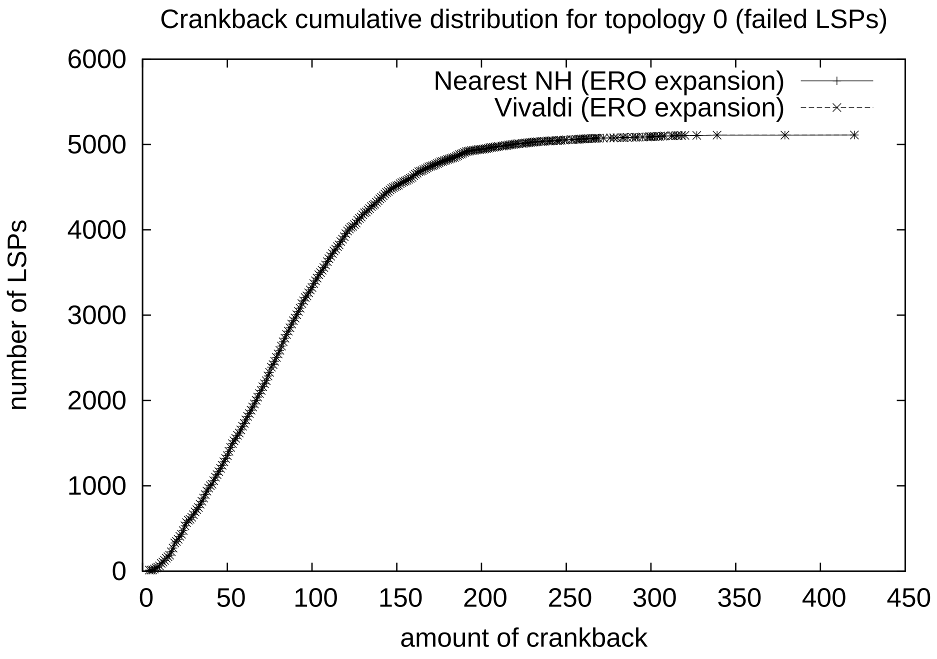


Figure 6.9: Amount of crankback for LSPs that cannot be established on topology with 20 transit ASs

tive PCEs. The global PCE does not rely on BGP. With the global PCE, 100% of the LSP can be established. 94% of the LSPs can be established with cooperative PCEs and 92% of the LSPs can be established along IP forwarding paths.

We make the same observations on the topology composed of 20 transit ASs, in figure 6.11. That is, more paths with a low delay are established with the global PCE than with a complete exploration of the downstream PCEs in cooperative PCEs. This is due to BGP route distribution and filtering. In addition, the number of paths with a low delay is larger for cooperative PCEs than for IP forwarding. With the global PCE, 100% of the LSP can be established. 94% of the LSPs can be established with cooperative PCEs and 84% of the LSPs can be established along IP forwarding paths, on the topology with 20 transit ASs.

Figure 6.12 shows the number of LSPs for which a path respecting the constraints can be found by each different setting of cooperative PCEs. At the top, the cooperative PCEs used “nearest NH ” to rank the NHs and contacted only a limited number of PCEs with domains containing the best NHs. At the bottom, we have the LSPs that are computed with the vivaldi heuristic. We consider the topologies composed of 5 transit ASs. The bars labeled “Total” represent the number of LSPs to be established on each topology. It is also the number of LSPs that are established with CSPF by the global PCE. Then, we have bars for the number of LSPs

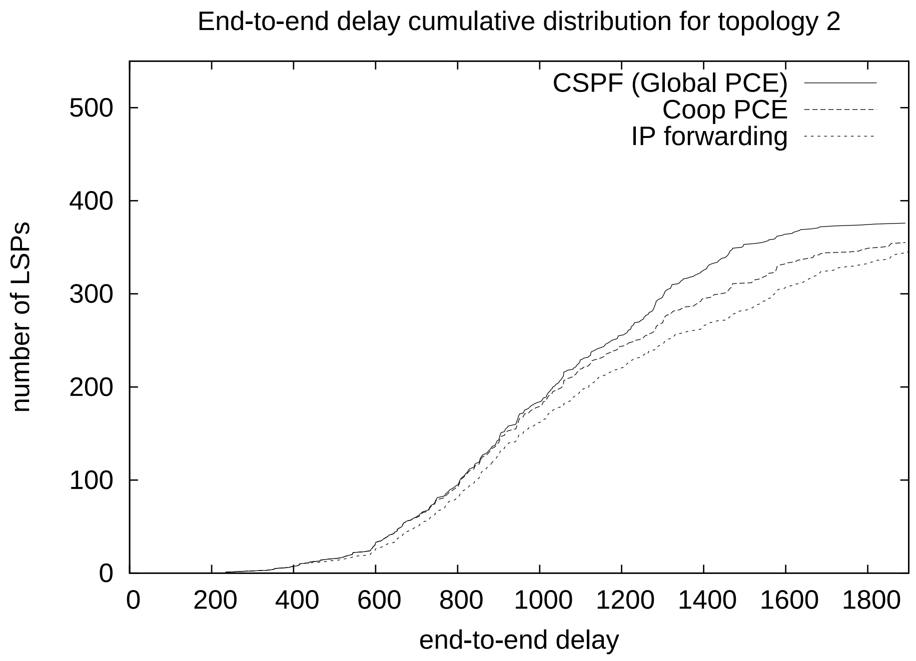


Figure 6.10: Path quality with cooperative PCEs on topology with 5 transit ASs

computed with a complete exploration of the downstream PCEs. The other bars show the number of LSPs that can be established by cooperative PCEs with a limit on the number of downstream PCEs that are contacted by each PCE participating in the computation.

At the top of figure 6.12, we see that a lot of LSPs cannot be established when a single downstream PCE is contacted per PCE. With this setting of cooperative PCEs, only 48% of the LSPs are established on topo0, 68% on topo1, 56% on topo2, 58% on topo3 and 58% on topo4. We note that more LSPs are established when the single PCE to be contacted is selected based on the vivaldi heuristic (bottom of figure 6.12). In this case, there are 89% of LSPs established on topo0, 93% on topo1, 78% on topo2, 86% on topo3 and 88% on topo4. Thus, more than 20% of additional LSPs are established when the downstream PCE selection is made with vivaldi.

We observe in figure 6.12 that when each PCE is allowed to contact two downstream PCEs, a lot more LSPs can be established for both heuristics, on each topology. With nearest NH, an average of 95% of the traffic demand is supported. An average of 96% of the LSPs are established with vivaldi.

Now, we compare the end-to-end delay and the number of LSPs that can be established when limiting the number of downstream PCEs that are contacted by a PCE. Figure 6.13 presents the results for a topology composed of 5 transit ASs.

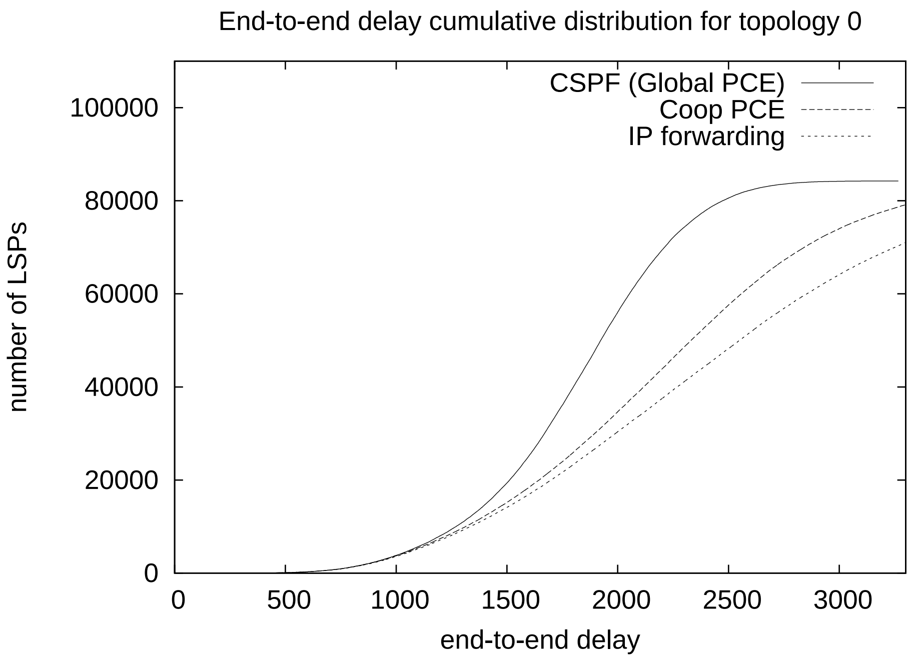
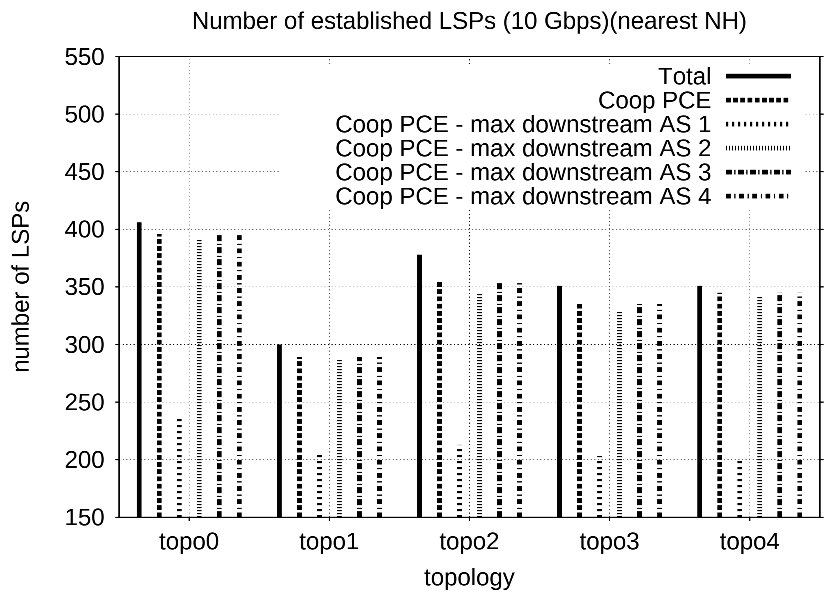


Figure 6.11: Path quality with cooperative PCEs on topology with 20 transit ASs

Again, we note that the results obtained on the four other topologies are similar. In figure 6.13, we have a curve with the LSPs established by the global PCE and a curve for the LSPs obtained with a complete exploration by means of cooperative PCEs. The other curves show the delay of the paths established when the PCEs contact 1, 2 and 3 downstream PCEs, respectively. At the top, we have the results with the nearest NH heuristic. At the bottom, the vivaldi heuristic is used to determine the downstream PCEs that are contacted.

For each heuristic, we note that less LSPs are established when 1 downstream PCE is contacted than when 2, 3 and all available downstream PCEs are contacted (Figure 6.13). Moreover, a limit of two downstream PCEs gives good results. Such a limit largely improves the results obtained with a limit of 1 downstream. Additionally, this curve approaches the results achieved by a complete exploration of the downstream PCEs. Finally, we observe that the results are the same when there is no restriction on the number of contacted PCEs as when a limit of 4 PCEs is set.

Comparing the heuristics (Figure 6.13), we note that vivaldi is better than nearest NH in selecting downstream PCEs to contact. With the same limit on the number of downstream PCEs, more LSPs can be established with vivaldi than with nearest NH. With a limit of one downstream PCE, 57% of the LSPs can be established, with nearest NH, and 78% of the LSPs with vivaldi. When the limit is



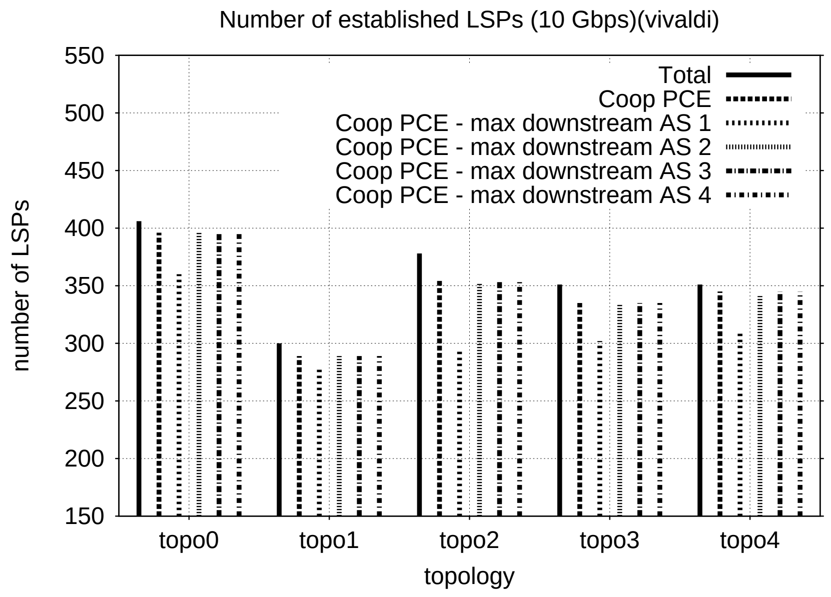
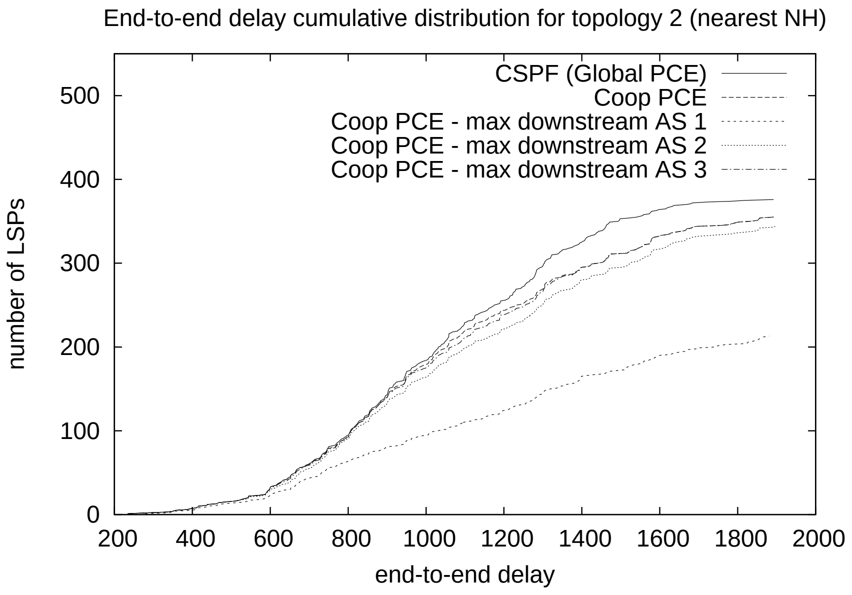


Figure 6.12: Number of LSPs established with cooperative PCEs on topologies with 5 transit ASs

raised to 2 downstream PCEs, 91% of the LSPs can be established with nearest NH and 93% with vivaldi, the maximum percentage of the LSPs established with a complete exploration being 94%.

On the topology composed of 20 transit ASs (Figure 6.14), we also observe



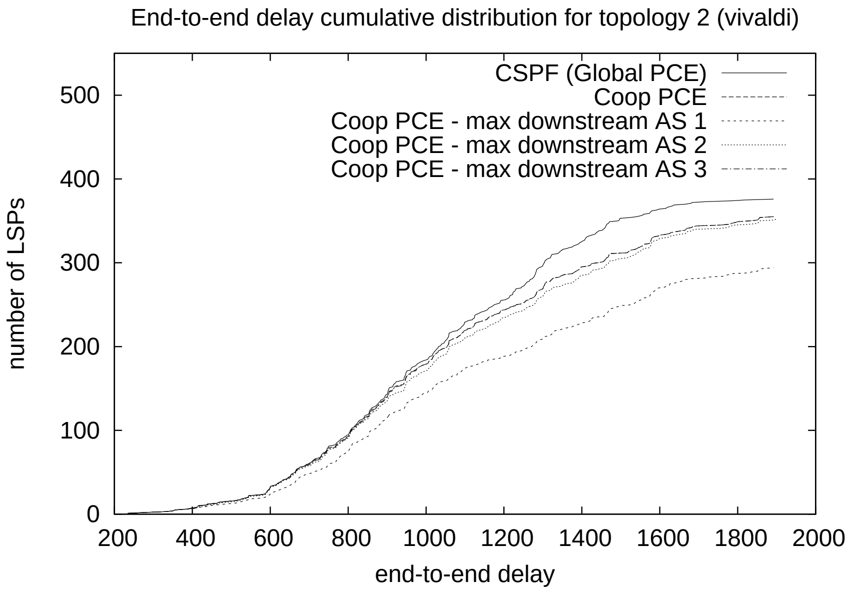


Figure 6.13: Path quality with cooperative PCEs and the heuristics on topology with 5 transit ASs

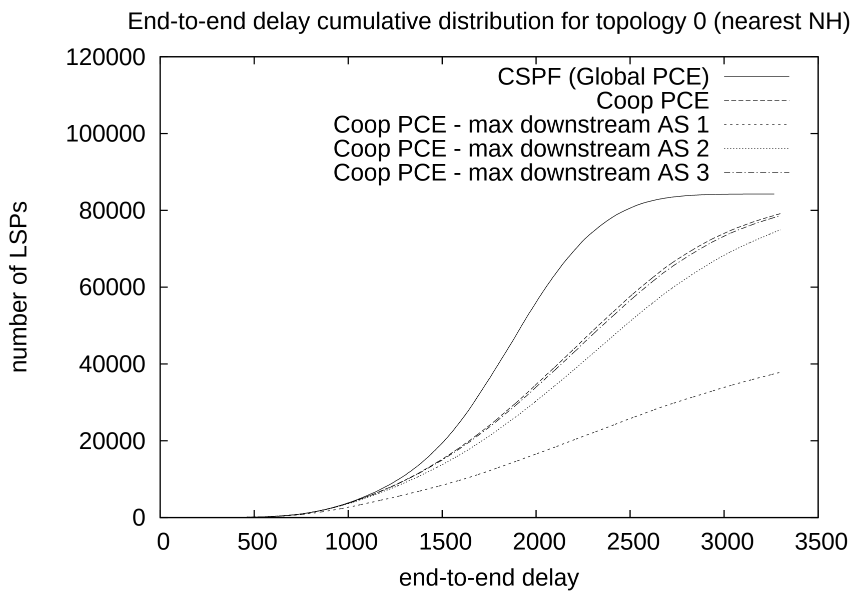
that the vivaldi heuristic gives better results than nearest NH for a downstream exploration limit of 1. With a limit of 1, nearest NH finds a suitable path for 45% of the LSPs while vivaldi finds a path for 61% of the LSPs. The difference between the two heuristics is not significant for an exploration limit of 2, 3 and 4. A limit of 2 already enables to improve largely the amount of LSPs that can be established compared to a limit of 1 downstream PCE. The percentage of established LSPs with a limit of two is equal to 89%, with both heuristics. A limit of 3 downstream PCEs enables to establish 93% of the LSPs. This is only one percent below the percentage of LSPs established with a complete exploration.

During the computation of paths by means of cooperative PCEs, PCReq and PCRep messages are exchanged. In our simulations, we computed upper and lower bounds on the number of these messages. The algorithms used for the computation of these bounds are provided in section A.4 of appendix A. The upper bound of the number of messages exchanged for the computation of an LSP represents the maximum number of messages that can be exchanged between PCEs for the computation of its path, with a given computation technique. This bound can be achieved if all the PCEs that participate in the computation of the LSP do not maintain state from previous computations of the same LSP. For this bound to be reached, a PCE never receives a PCRep for an LSP for which it is already treating a PCReq[6]. The lower bound on the number of PCReq and PCRep messages for an LSP gives an idea of the minimum number of messages that has to be exchanged in order to find a path for the LSP. It is equal to twice the number of AS adjacencies that are used to carry PCReq and PCRep messages in the computation of the path. Thus, in the lower bound, we do not consider that a PCE may contact the same downstream PCE multiple times, asking for a path that starts at a different ingress ASBR than earlier. However, this could happen in the cooperative PCEs computation technique that we model in our simulator (see appendix A). This lower bound may also not be reached by the BRPC technique described in [VZB06]. In BRPC, if there is no local path segment that respects the constraints of the LSP, the downstream PCEs are still contacted. The lower bound that we compute doesn't take into account these messages. Consequently, our lower bound is a lower bound for both our model of cooperative PCEs and BRPC. The lower and upper bounds provide an estimation of the number of PCReq and PCRep messages exchanged with stateful and stateless PCEs respectively.

Figures 6.15 and 6.16 show the number of PCReq and PCRep messages that are exchanged during the computation of LSPs that can be established on a topology with 5 transit ASs. At the top, we have the number of messages exchanged with the nearest NH heuristic. At the bottom, we have the number of messages exchanged with vivaldi. Figure 6.15 shows the upper bound on the number of messages for each heuristic. Figure 6.16 gives the lower bounds on the number of messages exchanged. We have a curve for each limit on the downstream PCEs that are contacted by a PCE.

We see on figure 6.15 that the vivaldi heuristic is slightly better than nearest

[6]If a PCE receives a second PCReq while a PCReq is currently treated by the PCE for the same LSP, the PCE still has a state for the LSP. The same state may be used for the two PCReq. Thus, the PCE may not need to contact all the downstream PCEs that have already been contacted for the PCReq under treatment again for the new request (see appendix A). Such a situation enables PCEs to exchange less messages than the computed upper bound.



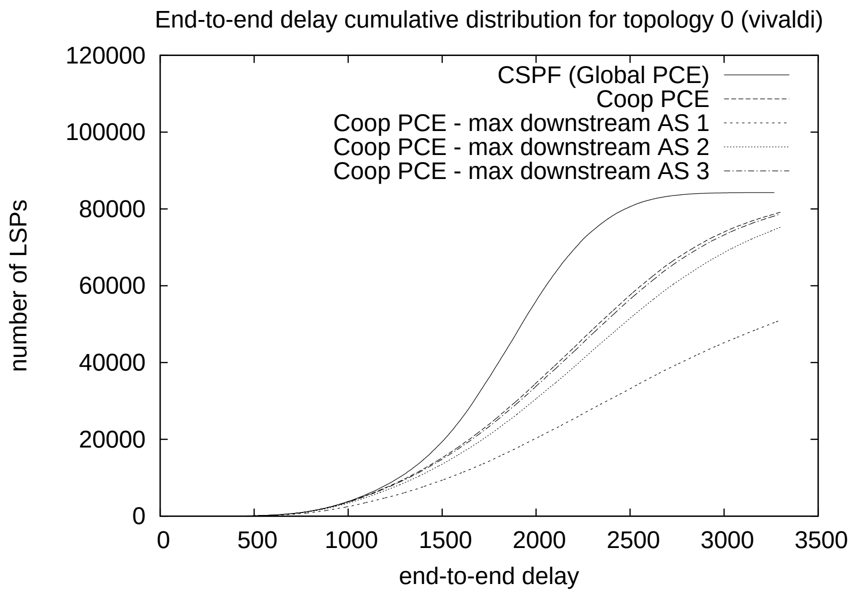
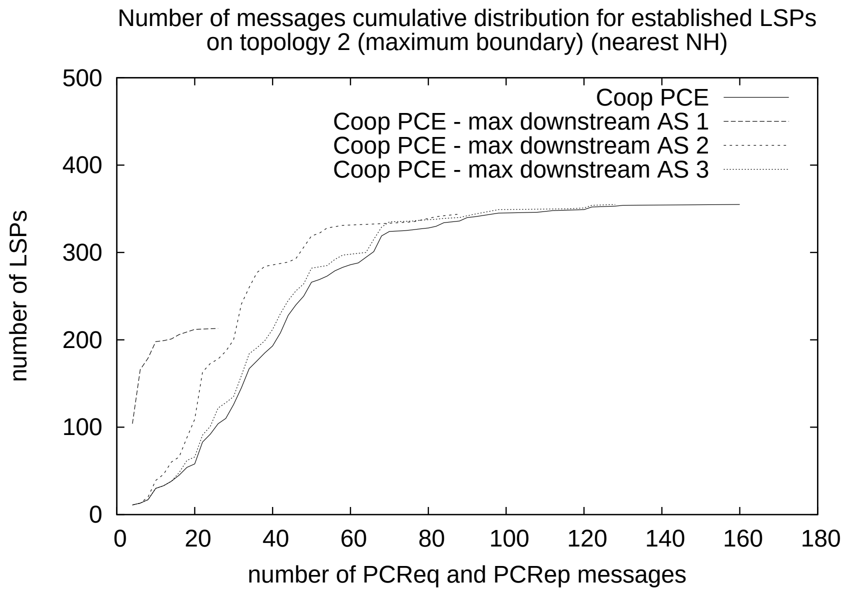


Figure 6.14: Path quality with cooperative PCEs and the heuristics on topology with 20 transit ASs

NH. When considering the upper bound on the number of messages, 26 is the maximum number of messages exchanged between PCEs with a limit of 1 downstream PCE and the nearest NH heuristic. With vivaldi, the maximum number of messages exchanged between PCEs is 22, with a limit of one downstream PCE.



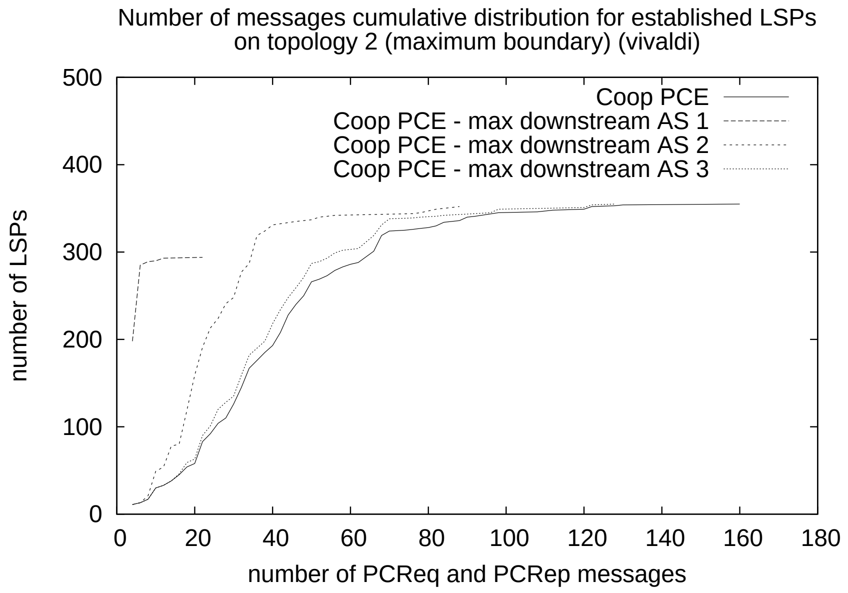
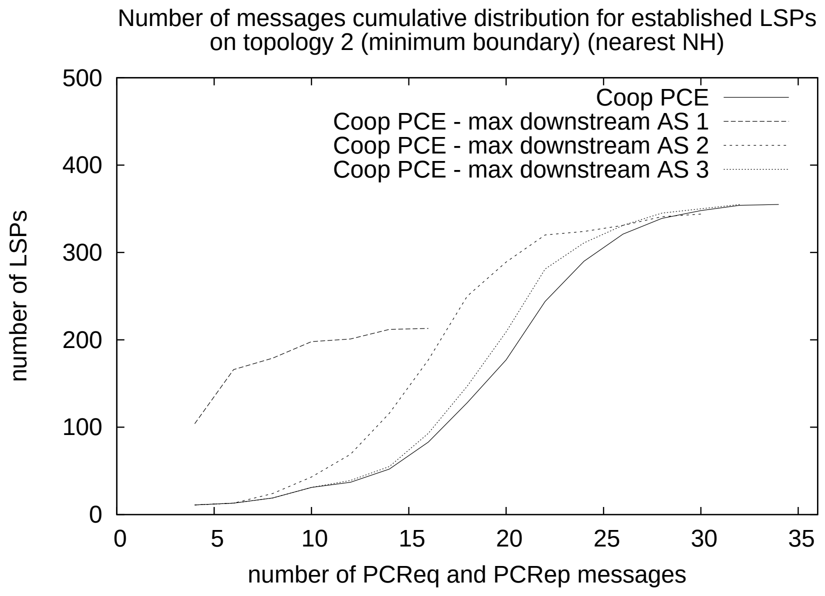


Figure 6.15: Signaling overhead with cooperative PCEs and the heuristics on topology with 5 transit ASs (upper bound)

The maximum number of messages exchanged between PCEs is the same for both heuristics when the limit of downstream PCEs that can be contacted is 2, 3 and 4. The maximum number of messages that are exchanged in the computation of LSPs with these limits is 88, 128 and 160, respectively. However, we note that there are



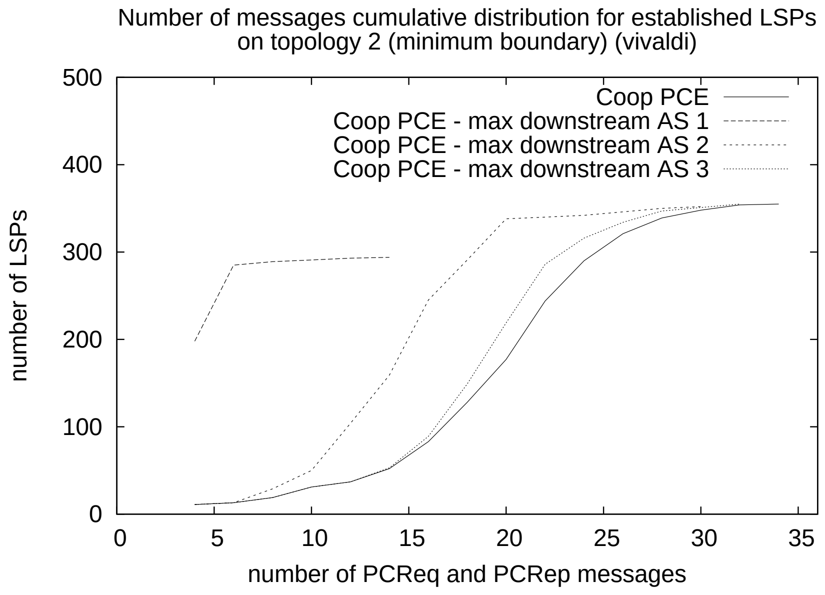


Figure 6.16: Signaling overhead with cooperative PCEs and the heuristics on topology with 5 transit ASs (lower bound)

more LSPs requiring less messages in the computation of their paths with vivaldi than with nearest NH.

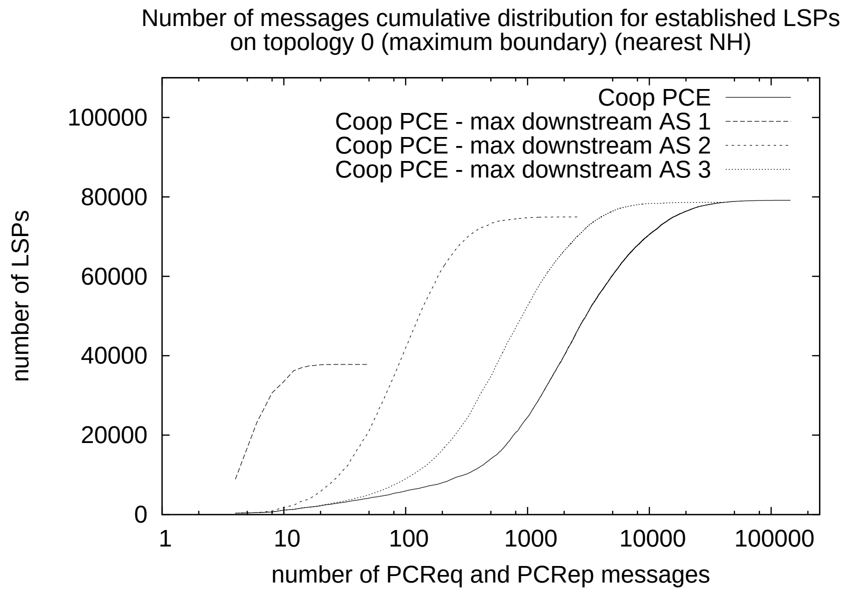
The same observation is made for the lower bound on the number of messages exchanged between PCEs (see figure 6.16). The vivaldi heuristic performs slightly better. The difference between the two heuristics is higher when the limit on the downstream PCEs that are contacted by a PCE is low. With a limit of 1, there are at most 16 messages that are exchanged with nearest NH, for the lower bound on the number of exchanged messages. This number is 14 with vivaldi. When the limit on the number of downstream PCEs that can be contacted increases, this number becomes the same for both heuristics. It is 30 for a limit of 2, 32 for a limit of 3 and 34 for a complete exploration. However, we also note, as in the upper bound case, that the lower bound on the number of messages is lower with vivaldi than with nearest NH for the computation of more LSPs.

Figures 6.17 and 6.18 show upper and lower bounds on the number of messages exchanged in the computation of the LSPs on the topology with 20 transit ASs. In these figures, a logarithmic scale in base 10 is used on the x-axis. At the top, we have the results for the nearest NH heuristic. At the bottom, we have the results for the vivaldi heuristic. The upper bounds on the number of exchanged messages are presented in figure 6.17 while the lower bounds are presented in figure 6.18.

We observe in these figures (6.17 and 6.18) that the difference between vivaldi and nearest NH is more significant on the topology with 20 transit ASs than on the smaller topologies (5 transit ASs). With nearest NH, a maximum of 48 PCReq and PCRep messages are exchanged in the computation of the LSPs, when a single downstream PCE is contacted by each PCE (top of figure 6.17). With vivaldi, this number is lower it is equal to 38 (bottom of figure 6.17). With a limit on 2 contacted downstream PCEs per PCE, we have a maximum of 2700 messages, with nearest NH, and 1134 messages, with vivaldi. When 3 downstream PCEs are contacted by a PCE, we have a maximum of 38672 messages with nearest NH and 17376 messages with vivaldi. With a limit of 2 and 3 downstream, the maximum number of messages exchanged is more than two times larger with nearest NH than with vivaldi. Finally, with a complete exploration, there are LSPs that require up to 144000 PCReq and PCRep messages.

The difference between the nearest NH and vivaldi heuristics is smaller when we consider the lower bounds on the number of PCReq and PCRep messages (figure 6.18). As observed for the topology composed of 5 transit ASs, we still note that the lower bounds with vivaldi are below the lower bounds with the nearest NH heuristics, on the topology with 20 transit ASs. The highest value of the lower bound of messages exchanged with a limit of one contacted downstream PCE requires is 32 messages with nearest NH. It is 24 with vivaldi. This highest value is 98, with a limit of 2 downstream, and 124, with a limit of three downstream with nearest NH. The highest value on the lower bound of messages exchanged by a complete exploration is 144 messages. That is, there is one LSP that requires at least 144 messages for the computation of its path with a complete exploration on the downstream PCEs at each PCE. With vivaldi the maximum number of messages on the lower bounds for an exploration of 2 and 3 downstream PCEs are 82 and 120. This is slightly better than with the nearest NH heuristic.

As a conclusion on path computation by means of cooperative PCEs, we can say that an exploration of one downstream PCE per PCE does not allow to find a



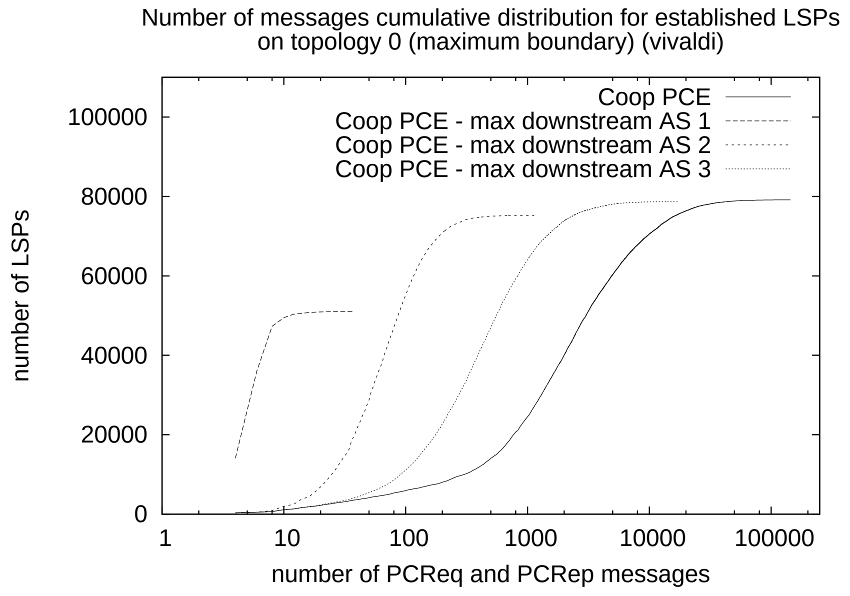
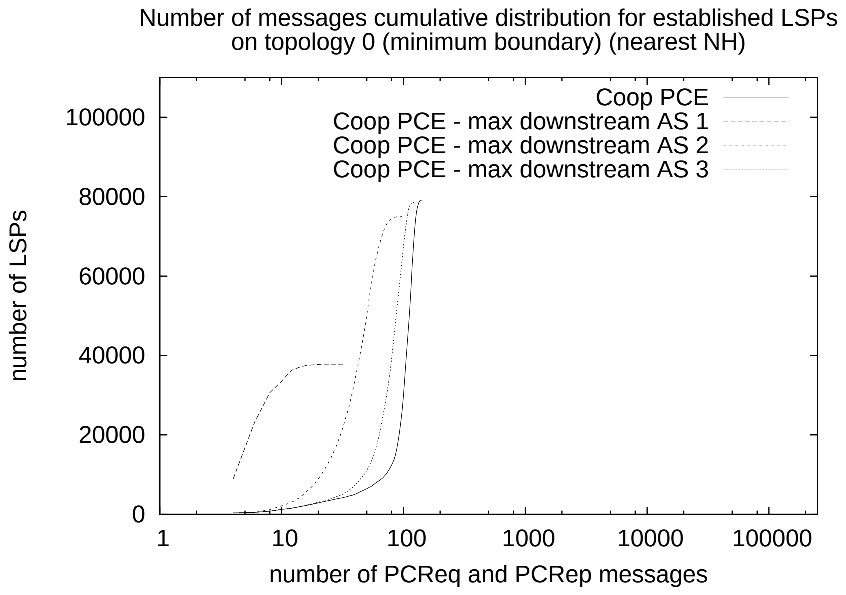


Figure 6.17: Signaling overhead with cooperative PCEs and the heuristics on topology with 20 transit ASs (upper bound)

good percentage of the paths. We observed that with an exploration of two downstream PCEs, the amount of LSPs with a suitable path increases significantly. A wider exploration allows to find more paths. However, the gain is reduced. Moreover, the increment in number of messages between an exploration of 2 to an ex-



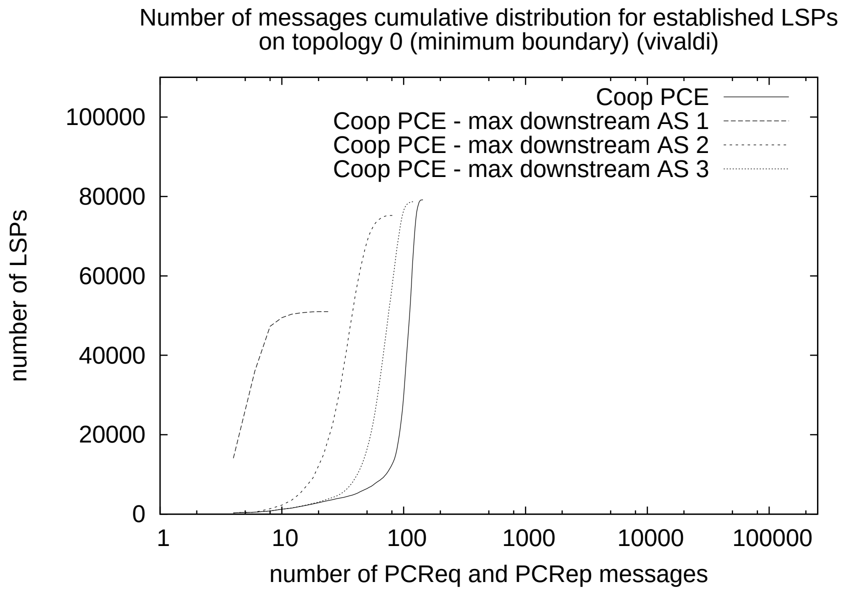


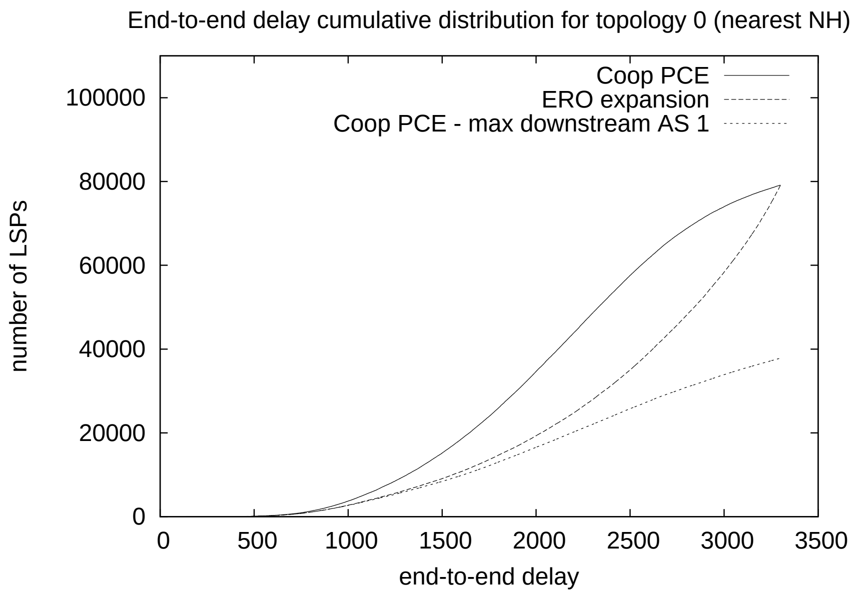
Figure 6.18: Signaling overhead with cooperative PCEs and the heuristics on topology with 20 transit ASs (lower bound)

ploration of three downstream PCEs is very high when PCEs are stateless (upper bound), especially on the large topology. This increment is smaller but significant when we consider the lower bound on the number of messages. Thus, an exploration of two seems to be a good compromise. Moreover, these results do not seem to depend tightly on the number of ASs that are connected to the stub and transit ASs. In the small topologies, stub ASs have up to 4 providers. In topo 2, the average number of providers is 2.4. Transit ASs are connected to at most 4 other transit ASs in topo 2, composed of 5 transit ASs. The average number of transit ASs directly connected to a transit AS is 2. Because stub ASs do not advertise routes for prefix that they don't own, only the PCE in the stub AS of the destination may be contacted. The connections with the other stubs do not have to be considered. Thus, for the small topologies there is no wonder that restraining the computation to two downstream PCEs provides good results. However, inter-AS connectivity is different for the large topology composed of 20 transit ASs. In this topology, the stubs have at most 6 providers with an average of 2.7 providers per stub. A transit is connected to at most 7 other transit ASs. Moreover, the average number of directly connected transit ASs to a transit AS is 4.2. We deduce that even with an increased connectivity, the number of downstream PCEs involved in cooperative PCEs does not need to be increased to a higher value than 2, in order to obtain good results. Exploring 2 downstream PCEs is a good compromise between the number of LSPs established and the signalling overhead, when we compare with the results obtained for larger explorations. Further studies may be required to determine if this is still true with a much larger increase in inter-AS connectivity.

6.4 Comparison of the Path Computation Techniques

In this section, we compare the ERO expansion technique to the computation by means of cooperative PCEs. We compare the quality of the paths, the number of LSPs established and the number of messages sent to and received from PCEs. For this comparison, we consider the topology composed on 20 transit ASs. We have seen in the two previous sections that the other topologies give analogous results even if these results are less contrasted on the small topologies. We compare ERO expansion to a complete exploration by means of PCE as well as to an exploration that is limited to one downstream PCE for each PCE. As in the previous sections, for each criterion, we have one figure with the techniques making use of nearest NH and one figure with the techniques that use the vivaldi heuristic.

Figure 6.19 shows the cumulative distribution of the end-to-end delay for the LSPs computed with the different techniques. For both heuristics, we see that there are more LSPs with a low delay with a complete exploration performed by cooperative PCEs. As we already said, this technique computes the shortest delay path that can be found based on the BGP routes. Then, for both heuristics, there are more LSPs with a low delay with ERO expansion than with cooperative PCEs that only contact one downstream PCE. Less paths respecting the delay constraint of the LSPs are found with a limit of one downstream PCE for cooperative PCEs. However, if we look back at figure 6.14, we observe that an exploration of two downstream PCEs is better than ERO expansion to find short delay LSPs. In figure 6.19 the curves for ERO expansion stick to the curve with cooperative PCEs with a limit of 1 downstream for short delays. However, in figure 6.14, cooperative PCEs find more short delay paths when using two downstream ASs than when using one.



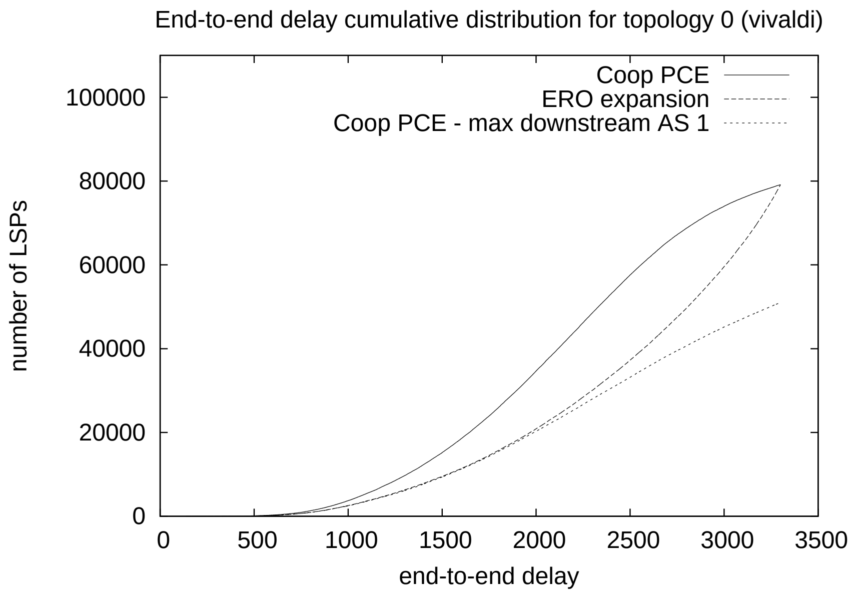


Figure 6.19: Path quality with cooperative PCEs and ERO expansion on topology with 20 transit ASs

Finally, we note from figure 6.19 that even if the paths obtained by the ERO expansion heuristics have a higher delay than the paths computed by cooperative

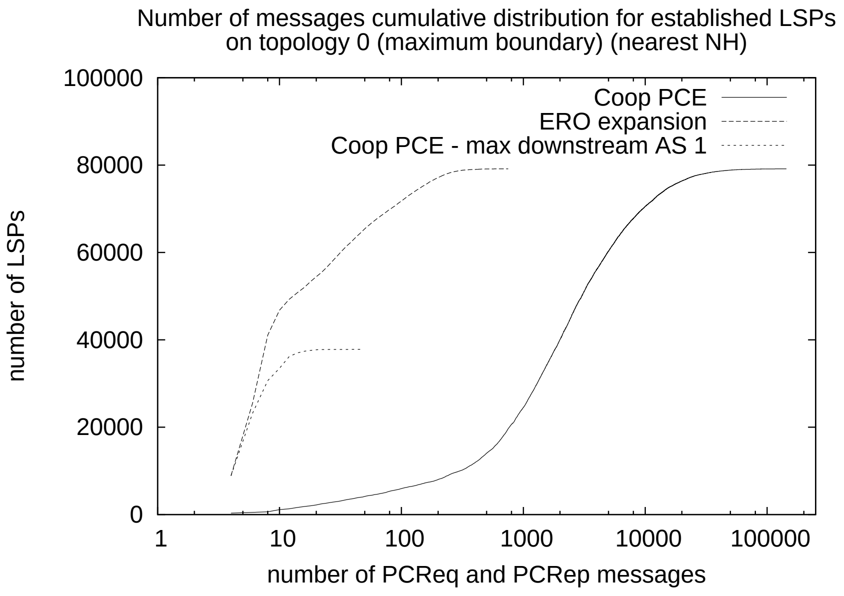
PCEs, ERO expansion enables to find as many paths respecting the maximum end-to-end delay of the LSPs as a complete exploration with cooperative PCEs. Cooperative PCEs return the path with the lowest delay that is available from the BGP routes. However, in the ERO expansion technique an LSP establishment is terminated as soon as a path respecting the constraints is found even if a better path exists.

Figures 6.20 and 6.21 show the cumulative distribution of the number of PCReq and PCRep messages generated by each technique. We use a logarithmic scale in base 10 on the x-axis.

For the techniques relying on cooperative PCEs, that is the curves labeled "Coop PCE" and "Coop PCE - max downstream AS 1", in figures 6.20 and 6.21, we consider upper and lower bounds on the number of PCReq and PCRep messages required. With these techniques the amount of messages that are exchanged depends on the type of PCEs (stateless or stateful) participating in the computation and the timing of the messages, if the PCEs are stateless. Thus, in a deployed environment, the number of messages is comprised between the two boundaries provided for each technique.

In figures 6.20 and 6.21, we provide, for each heuristic, a single curve for ERO expansion. This curve presents the exact number of PCReq and PCRep messages required by our implementation. In our implementation of the ERO expansion technique, the PCEs are stateless. It is the PCCs that maintain state. They currently only keep the list of NHs that have been tried and led to infeasible paths as well as the delay of the path segment from the LSP's head-end to the PCC. This state is not to be compared with the one that is maintained inside stateful PCEs. It corresponds to the information that is required at the PCC to deal with an RSVP Path message. Each time the PCC receives a Path message for an LSP, PCReqs are sent to the PCE until a suitable NH and path segment is found or there is no more potential NH and path segment terminating at this NH along which the Path message has not been sent. It is for this purpose that a PCC has to remember the NHs that have been tried before. However, if there is no suitable NH at the PCC, crankback occurs and this state is removed from the PCC.

It is interesting to maintain state inside cooperative PCEs because once its downstream PCEs have been asked for path segments starting at given ingress ASBRs, the downstream paths from these ingress ASBRs to the destination are known to the PCE. In addition, the properties of these paths, that are relevant for the LSP, are also known at the PCE. Thus, the PCE can use this information for subsequent requests that it may receive for the same LSP. However, it is not interesting to maintain state inside PCEs that perform ERO expansion. With ERO expansion, the path computation through a node is stopped as soon as it cannot be completed because the path from the head-end LSR to the NHs of the current node do not respect the constraints of the LSP. This may be due to the properties of the path upstream to the current node. Thus, the properties of the possible paths downstream of the current node are not entirely known. If a new request for the LSP arrives at the same node, all the NHs have to be tried again because the downstream path may now be



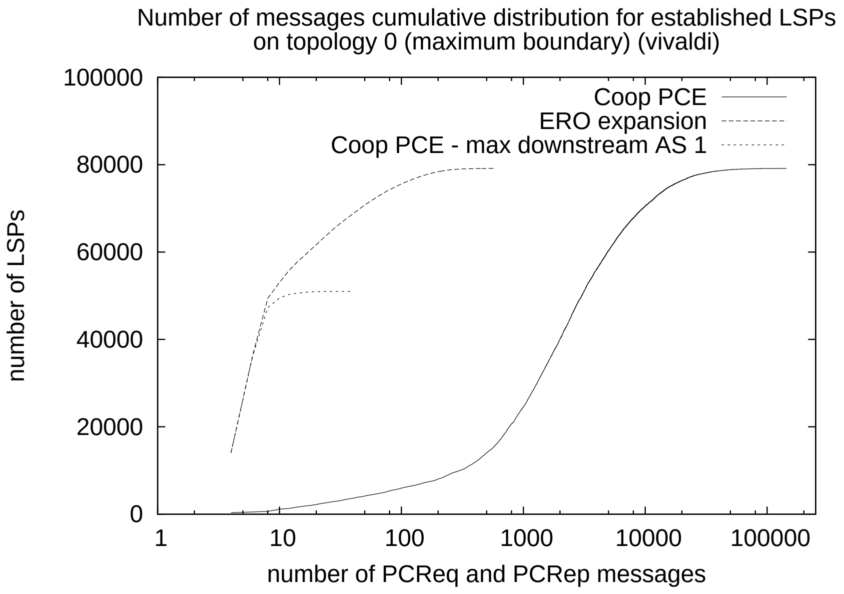
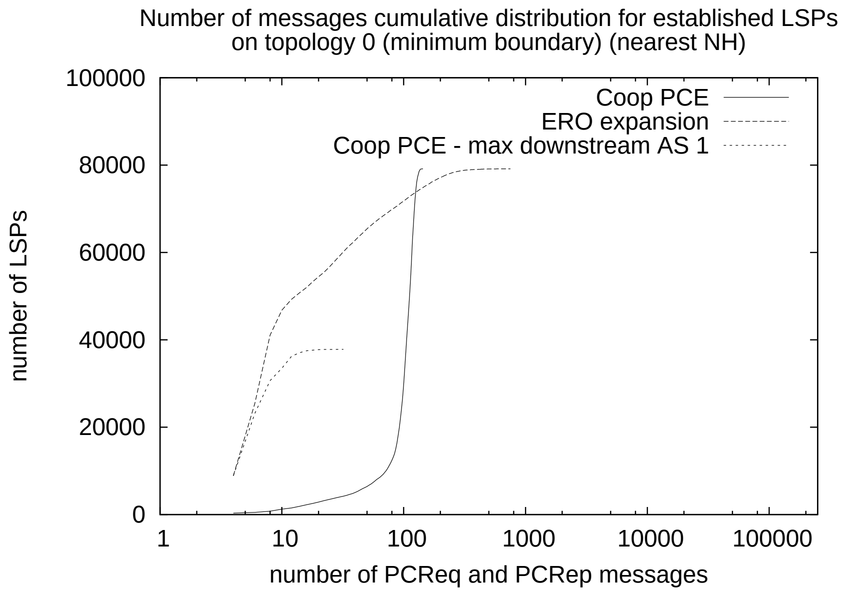


Figure 6.20: Signaling overhead with cooperative PCEs and ERO expansion on topology with 20 transit ASs (upper bound)

suitable with regard to the new upstream path that is used to join the current node. NHs cannot be pruned because a previous computation through them lead to an unsuccessful LSP establishment.

We see in figure 6.20 that for both heuristics, the number of PCReq and PCRep



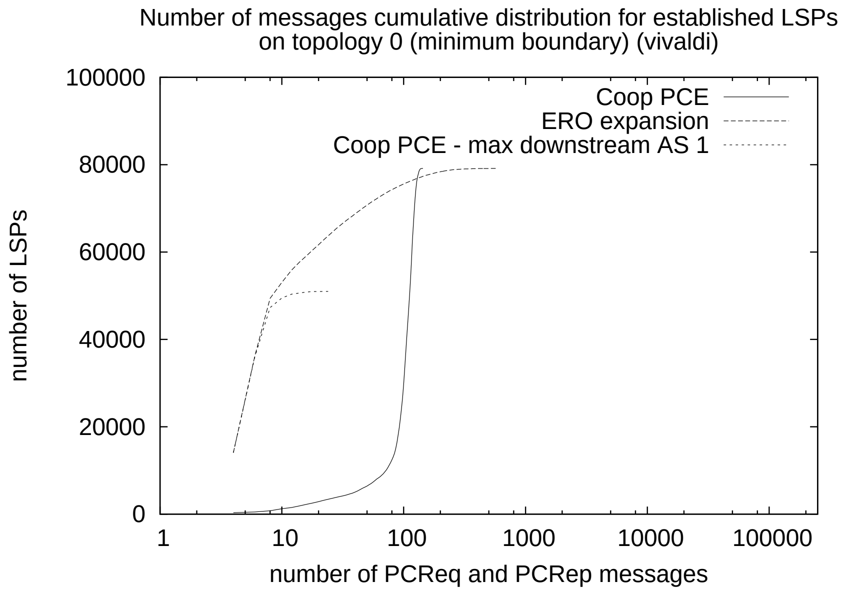


Figure 6.21: Signaling overhead with cooperative PCEs and ERO expansion on topology with 20 transit ASs (lower bound)

messages exchanged with ERO expansion is far below the maximum number of messages exchanged with a complete exploration by cooperative PCEs, on the upper bound curve. The maximum number of PCReq and PCRep messages exchanged with ERO expansion and the nearest NH heuristic is 752. With ERO expansion and vivaldi, this number is 590. The maximum number of messages on the upper bound curve for cooperative PCEs is 144000.

In addition, the maximum number of messages exchanged with ERO expansion (both heuristics) is higher than the maximum number of messages between cooperative PCEs, with a limit of one downstream PCEs that is contacted per PCE. However there are more LSPs established with ERO expansion that require fewer messages than cooperative PCEs with a limit of one downstream. This is true for the upper and lower bounds (figures 6.20 and 6.21) on the number of messages with a limit of 1 for the cooperative PCEs.

Third, the maximum number of messages exchanged with PCEs in ERO expansion (both heuristics) is below the number of messages exchanged between cooperative PCEs contacting at most two downstream PCEs.

Finally, we observe in figure 6.21 that there are LSPs that require more PCReq and PCRep messages with ERO expansion than the lower bound on the number of messages exchanged by a complete exploration with cooperative PCEs. However, we note that in order to achieve this lower bound state has to be maintained inside PCE while no state is maintained inside PCEs with ERO expansion.

The PCReq and PCRep are not the only messages that are required for the computation and the establishment of the LSPs. RSVP Path and Resv messages are required for the establishment of the LSPs by both cooperative PCEs and ERO expansion. The ERO expansion technique also requires the use of Path Error messages when crankback occurs. With ERO expansion, Path and Path Error messages are not only transmitted along the final path for the LSP, as it is the case for Path messages in cooperative PCEs (assuming that it is possible to establish the LSPs along the paths computed by the cooperative PCEs). These messages may cross nodes that belong to path segments that lead to an unfeasible path. Other alternatives were tried after crankback occurred. However, Path and Path error messages were treated at each node along all the path segments that lead to crankback. We do not count the RSVP messages together with the PCReq and PCRep messages because the processing of theses messages is different. The additional Path and Path Error messages exchanged in ERO expansion due to crankback have an impact on the routers. They are processed by the routers. On the contrary, PCEP messages can be treated by dedicated servers, the PCEs. The exchange of PCEP messages does not impact the routers except for their forwarding. This point should be taken into account in the choice between ERO expansion and cooperative PCEs.

In this analysis, we did not compare the state that is maintained by the different path computation techniques. State is maintained at different nodes depending on the computation technique. The amount of state also varies from one technique to the other. For cooperative PCEs, state may or may not be maintained inside PCEs after the completion of a reply message. In ERO expansion, state is not maintained inside PCEs but inside PCCs (ingress ASBRs, in our simulations). In addition to the acceptable amount of state to be maintained for the computation of constrained inter-AS LSPs, there is a choice to be made on where it is acceptable to maintain such state.

We have seen in this section that ERO expansion enables to compute as many LSPs respecting the delay constraint as a complete exploration with cooperative PCEs. In addition, the delay of the path computed with ERO expansion for an LSP is longer than the delay of the path obtained with a complete exploration with cooperative PCEs for the same LSP. However, there are a lot of LSPs that require less PCReq and PCRep messages with ERO expansion than the lower bound on the number of messages with a complete exploration by means of cooperative PCEs. In the previous section, we noticed that cooperative PCEs contacting at most two downstream PCEs was a good trade-off between the quality of the computed paths and the signalling overhead, for cooperative PCEs. In this section, we have shown that signalling between PCCs and PCEs in ERO expansion is lighter than the exchange of messages between stateless cooperative PCEs with a limit of two downstream PCEs.

Table 6.1 summarizes the evaluation of the path computation techniques performed in this chapter, for LSPs with a maximum end-to-end delay constraint. Each line in the table corresponds to a different path computation technique. The columns present the evaluation criteria. For example, line 1 of table 6.1 shows that we observed in this chapter the largest number of LSPs established with the global PCE computation technique. Moreover, there are many LSPs with a very low delay when the computation is performed by a global PCE. Finally, crankback does not occur and only two messages are exchanged between the PCC and the global PCE.

We see in table 6.1 that the technique with the lowest number of LSPs established is cooperative PCEs when a single downstream PCE that is contacted. The best technique regarding the number of LSPs established is global PCE. Global PCE is also the technique that provides the highest number of LSPs with low delays. Cooperative PCEs with a limit of one downstream PCE establish the fewest low delay LSPs. The only technique that requires crankback is ERO expansion. We have seen that the amount of crankback with ERO expansion is not too high. Concerning the number of messages exchanged between PCCs and PCEs, ERO expansion and cooperative PCEs with a limit of one downstream PCE perform well. The technique that requires the largest number of messages exchanged between PCCs and PCEs is a complete exploration with cooperative PCEs.

In table 6.1, we do not detail the results obtained for the different heuristics. We have seen in this chapter that the vivaldi heuristic performs better than nearest NH for both ERO expansion and cooperative PCEs. However, the results obtained for both heuristics are of the same order of magnitude when considering the same computation technique.

6.5 Conclusion

In this chapter, we presented two heuristics for the ranking of NHs in a domain. These heuristics aim at estimating the delay quality of the paths through each available NH for a destination. Then, we have shown how to use these two heuristics in

	Criteria			
	Estab LSPs	Low delay LSPs	Crankback	PCEP msg
Global PCE	very good	very good	–	excellent
IP forwarding	fair	good	–	–
ERO expansion	good	fair	good	very good
Cooperative PCEs (complete exploration)	good	good	–	bad
Cooperative PCEs - maxdownstream 1	bad	bad	–	very good
Cooperative PCEs - maxdownstream 2	good	good	–	fair

Table 6.1: Overview of the simulations results for delay constrained LSPs

the ERO expansion and cooperative PCEs path computation techniques.

We evaluated the use of the two proposed heuristics in the ERO expansion technique. We have seen that the vivaldi heuristic that relies on synthetic coordinates to estimate end-to-end delay performs better than nearest NH. The latter heuristics is only based on delay information that is local to the AS. We asked ourselves whether the use of vivaldi is relevant due to the additional computation and distribution of coordinated that is required.

Second, we looked at the performance of the complete exploration of the downstream PCEs with cooperative PCEs. We also studied the impact of the two proposed heuristics for selecting a subset of the downstream PCEs on the delay of the paths, the number of successful LSPs' establishment and the amount of signalling. Again, we showed that the vivaldi heuristic performs better than nearest NH. Moreover, we concluded that limiting the exploration of a PCE to two downstream PCEs is a good compromise in terms of LSPs established and signalling. This is valid for both the small and the large topologies.

Third, we compared the performance of ERO expansion and cooperative PCEs. In this analysis, we noted that the paths obtained with cooperative PCEs have lower delays than the paths computed with the ERO expansion heuristics. The former paths are the lowest delay paths that can be achieved based on the BGP routes. However, we said that cooperative PCEs either maintain states inside PCEs or exchange a lot of messages in order to compute these low delay paths. Less messages are exchanged for the computation of most constrained LSPs with the ERO expansion heuristics when a path respecting the constraint can be found than with cooperative PCEs. Moreover, the paths computed by these heuristics are not far from the best delays available to a computation method that relies on the BGP routes.

Chapter 7

LSPs with maximum delay and minimum bandwidth reservations

In chapter 6 we proposed heuristics to rank NHs for the computation of LSPs that are subject to maximum end-to-end delay constraints. Here, we first study the applicability of these heuristics for the computation of LSPs requiring bandwidth reservations in addition to the end-to-end delay constraint. Then, we analyze the interactions between our heuristics that contribute to the computation of minimum delay paths and the use of an intra-domain TE technique, called DAMOTE [BML03b], that aims at minimizing the traffic load inside the domains.

7.1 Evaluation of "nearest NH" and "vivaldi" Heuristics

The nearest NH and vivaldi heuristics were proposed in chapter 6 for the provision of paths with a maximum end-to-end delay constraint. In order to evaluate these heuristics for LSPs with a required bandwidth guarantee in addition to the delay constraint, we look at the amount of traffic that can be carried by each technique, with each heuristic. We also study the end-to-end delay of the paths that respect the required QoS (delay and bandwidth guarantees). For the ERO expansion computation technique, we look at the amount of crankback required to establish delay constrained LSPs with bandwidth reservations. Finally, we analyze the number of messages exchanged between cooperative PCEs with different limitations on the number of downstream PCEs that are contacted by each PCE.

In this section, we present the results of the simulations on the topologies introduced in section 6.3.1. We first describe the results obtained from the simulations with the topologies containing 5 transit ASs. Then, we analyze the results obtained on the larger topology, composed of 20 transit ASs.

For each topology, we performed several simulations. The link bandwidths are set to a different value in each simulation. The objective is to study the impact of various levels of congestion on the LSP's establishment techniques. In the first simulation, the bandwidth of all links is set to 10 Gbps. Then, it is set to 2400

Mbps in the second simulation. Finally, it equals 622 Mbps in the third simulation.

Each LSP is assigned a bandwidth reservation of 100 Mbps. With a bandwidth reservation of 100 Mbps we can emulate the Fast-Ethernet service between Service Providers. The delay constraints of the LSPs are the same as in section 6.3. LSPs to be established on topologies composed of 5 (20) transit ASs are subject to a maximum end-to-end delay constraint of 1900 units (3300 units, respectively).

In sections 7.1.1 and 7.1.2, we distinguish the LSPs for which a path respecting the constraints could be found in the topology, called “established LSPs”, from LSPs for which no suitable path could be found, called “failed LSPs”.

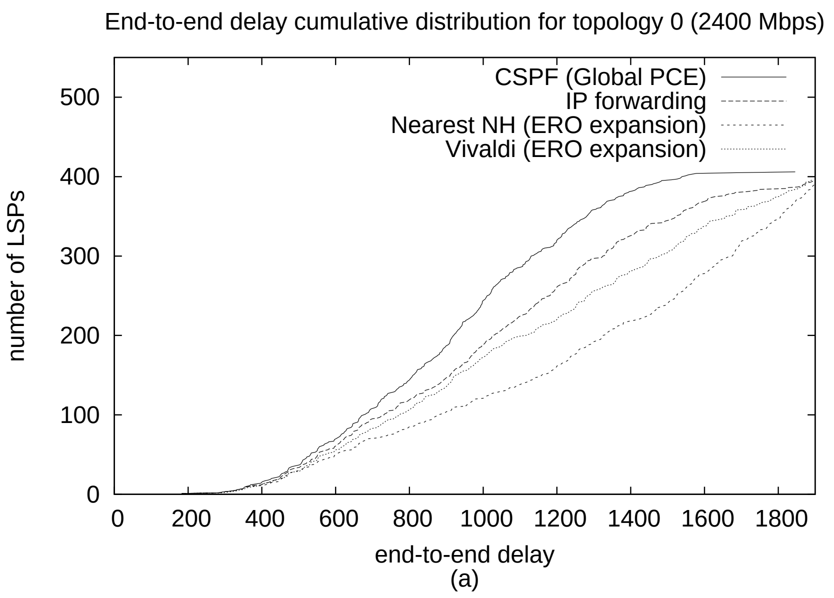
7.1.1 ERO expansion

In the analysis of the two heuristics with ERO expansion, we focus our attention on three aspects: the end-to-end delay of the LSPs, the number of LSPs that can be supported by the network, in our case this is proportional to the total amount of traffic that can be carried on the topology, and, finally, the amount of crankback that occurs during the computation of the constrained paths.

The curves in figures 7.1 and 7.2 are obtained from simulations on the small topologies with link bandwidths set to 2400 Mbps and 622 Mbps. The results from the simulations with 10 Gbps links are similar to the results obtained with link bandwidths equal to 2400 Mbps. This is because there is no congestion in the topology with 10 Gbps links and only a few links are congested with 2400 Mbps links. In the latter topology, only 2 links are congested with the ideal CSPF computation inside the global PCE. Moreover, 10 links are congested with the “nearest NH” heuristic and, 0 links with “vivaldi”.

Figures 7.1 (a) and 7.2 (a) show the cumulative distributions of the end-to-end delay for the different path computation techniques. They show, for a given delay on the x-axis, the number of established LSPs, on the y-axis, with end-to-end delay lower or equal to the value on the x-axis. Figures 7.1 (a) and 7.2 (a) present the results for a single topology, topology 0. Only the established LSPs are considered in these figures. The simulations performed with the other topologies with 5 transit ASs provide similar results.

In figure 7.1 (a), we first observe that there are more paths with a low delay with the CSPF computation performed by the global PCE than with the other techniques. We note that this computation does not rely on BGP. It is not constrained by BGP policies and filtering. It is used as a benchmark to which the other methods are compared. Moreover, there are more IP forwarding paths with a low delay than with the “nearest NH” and “vivaldi” heuristics. The good quality of the IP forwarding paths in terms of delay comes from the fact that many BGP routes in our simulation are selected based on the IGP cost. Since we set the IGP cost of a link to its delay, the BGP selection rule based on the IGP cost prefers a route with a low delay over a route with a longer delay. Finally, the “vivaldi” heuristic provides more paths with a low delay than the “nearest NH” heuristic. This is due to the fact that “nearest NH” selects the NH only based on delay information that is local



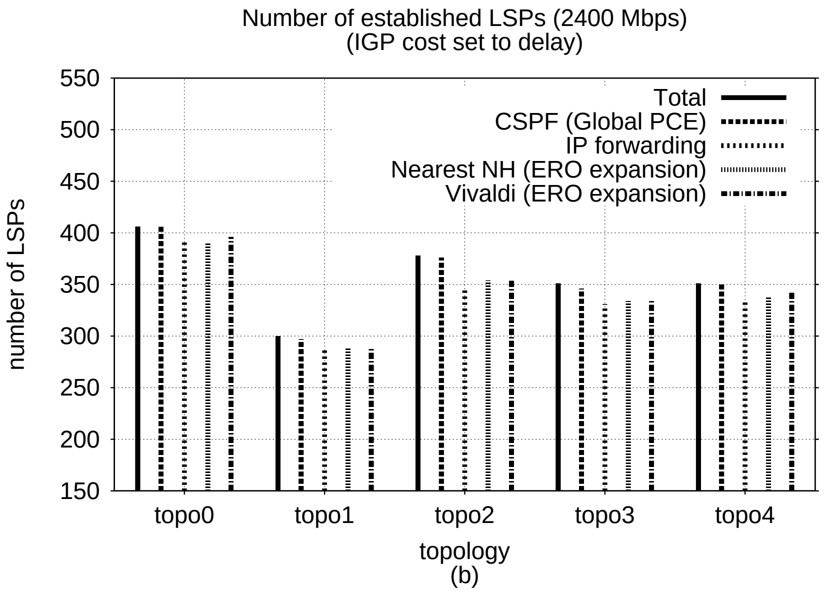
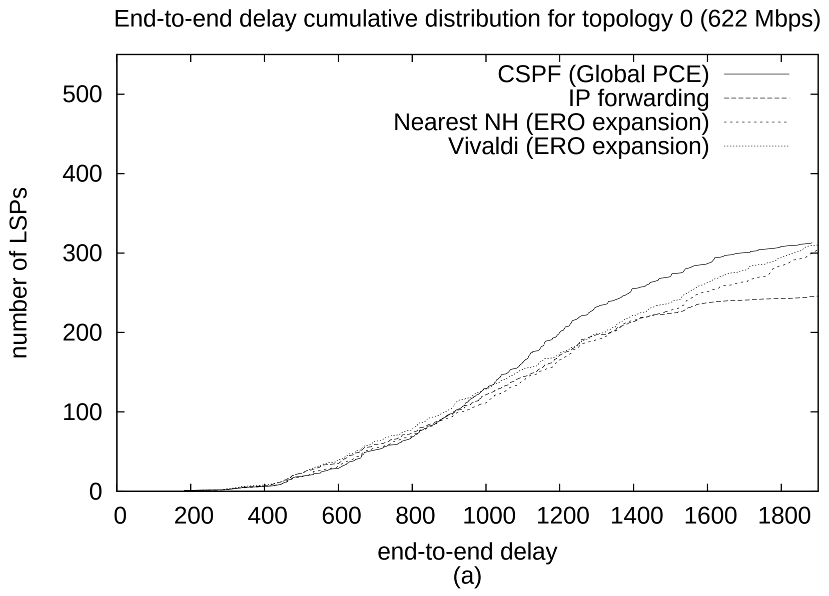


Figure 7.1: Delay of LSPs established on topology with 5 transit ASs (2400 Mbps)

to the domain whereas the "vivaldi" NH selection is based on an estimation of the delay of the path that transits through the candidate NH.

When the bandwidth of the links is set to 622 Mbps in topology 0, congestion occurs on 4% of the links with CSPF and on 3% of the links with "nearest NH" and "vivaldi" path computation techniques. We observe in figure 7.2 (a) that there is not much difference between the 4 curves for the LSPs with end-to-end delay below 1000 units. Above this value, there are more CSPF paths with a low



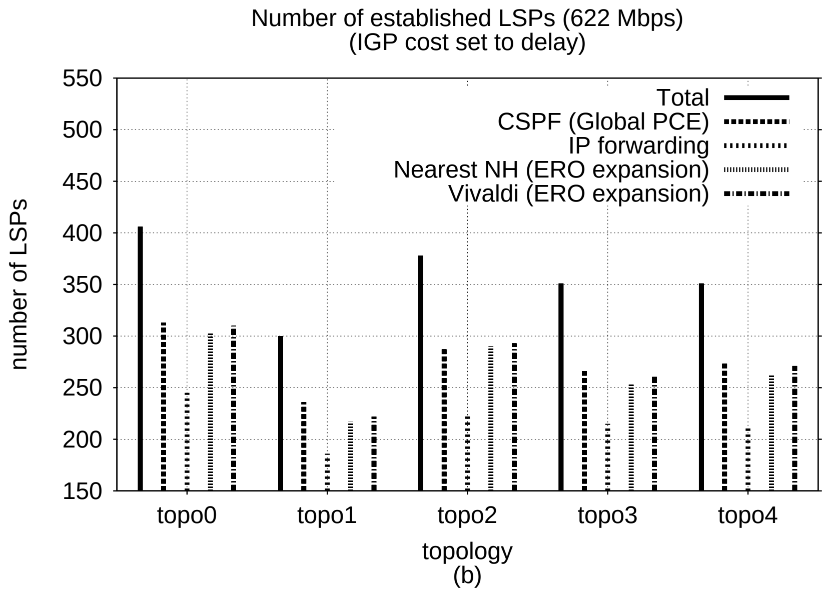


Figure 7.2: Delay of LSPs established on topology with 5 transit ASs (622 Mbps)

delay compared to the other path computation techniques. Finally, we note that the total number of LSPs established along the IP forwarding paths is below the number of LSPs established with CSPF and our two heuristics. With IP forwarding, a router can only use a few outgoing interfaces for a destination. When the corresponding links are congested, the router is not able to send the path establishment request on an alternate link. Thus, the LSP establishment fails. However, the "nearest NH" and "vivaldi" heuristics rely on RSVP-TE for the establishment of the LSPs. RSVP-TE enables to avoid congested links in the establishment of an LSP by specifying the path to be followed by the LSP inside the Explicit Route Object (ERO) in order to bypass IP forwarding. Thus, techniques based on ERO expansion are more robust to congestion than standard IP forwarding. At last, we see that there are slightly more paths with a low delay with the "vivaldi" heuristic than with "nearest NH". However this difference is not significant.

Figures 7.1 (b) and 7.2 (b) show the number of LSPs that can be successfully established on each topology, with the different path computation techniques. In figures 7.1 (b) and 7.2 (b), we observe that the number of LSPs established by the techniques relying on BGP routes, that is IP forwarding, "nearest NH" and "vivaldi", is lower than with CSPF. The CSPF paths are computed by a centralized entity that possesses the complete topology. With BGP, however, only a portion of the routes available for a destination is distributed. Since there are fewer routes, they become congested faster. Moreover, some of these routes are selected by BGP based on other criteria than the delay. Thus, the resulting paths learned for a destination do not necessarily have a lower delay than the maximum end-to-end delay constraint of the LSPs.

We note that the number of LSPs established with the "vivaldi" heuristic is slightly higher than this number for the simulations with "nearest NH". In both cases, the set of potential NHs depends on the BGP routes received for the destination. The set of potential NHs, inside an AS, is the same for many destinations. Among this set, the selection of the NH for a given ingress ASBR only relies on the delay of the shortest delay path with enough bandwidth for the LSP, in the "nearest NH" heuristic. Thus, the LSPs entering an AS through an ingress point incur the same delay until the shortest delay path becomes congested and a longer delay path is followed in the AS. Therefore, the delay incurred inside an AS by the LSPs entering at the same ingress ASBR increases as the LSPs are established. On the other hand, the selection of the NH by the "vivaldi" heuristic relies on the shortest delay path inside the AS and on the delay estimation from the NH to the destination of the LSP. Consequently, the delay incurred by the LSPs crossing an AS does not increase as fast as with the "nearest NH" heuristic because the LSP establishment requests entering an AS at an ingress point are distributed among multiple paths inside the AS based on the destination of the LSP. The delay of the paths computed with the "nearest NH" heuristic increases faster than with "vivaldi". As a consequence, if all LSPs are subject to the same delay constraint, this constraint will be harder to fulfill with "nearest NH" than with "vivaldi", as the LSPs are established.

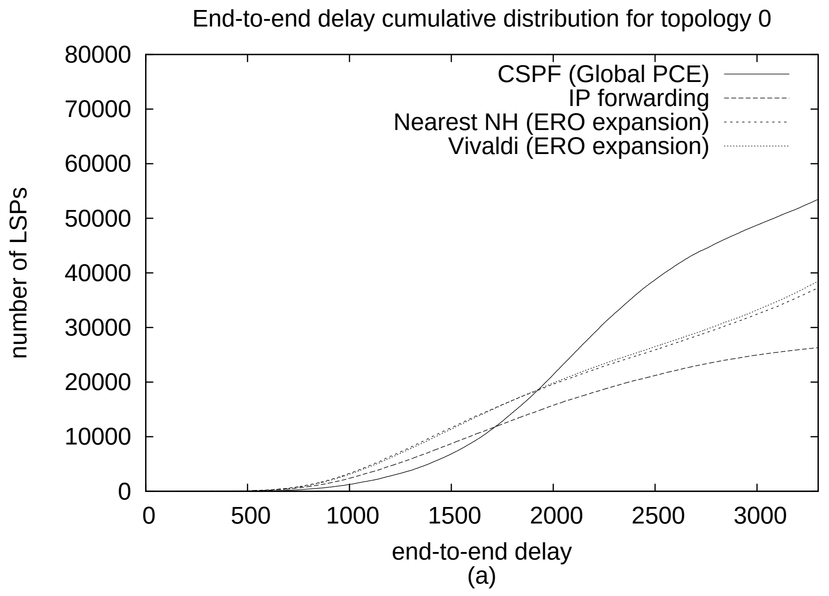
Figure 7.2 (b) shows that in a less provisioned network, compared to the results in figure 7.1 (b), the number of LSPs that can be established along IP forwarding paths drops, as mentioned earlier. In addition, the number of LSPs established with our two ERO expansion heuristics is not far below the number of LSPs established along the CSPF paths computed by the global PCE. This is mostly due to the use of the RSVP-TE ERO expansion technique and the crankback mechanism.

The crankback mechanism enables to inform an upstream node of the failure during the establishment of an LSP and to try the establishment of the LSP along another path. Crankback occurs at most 12 times with “vivaldi” and 25 times with “nearest NH” for established LSPs, on topology 0 with link bandwidths set to 2400 Mbps. Additionally, there are 95% of LSPs established with “vivaldi” without the help of crankback and 73% with “nearest NH”. On topology 0 with 622 Mbps links, there are 85% and 67% of the LSPs that are established without performing crankback with the “vivaldi” and “nearest NH” heuristics, respectively. Moreover, the maximum number of crankback for established LSPs is 13 for “vivaldi” and 14 for “nearest NH”. We observe that, for the congested topology, crankback enables to carry more traffic inside the congested topology than when the IP forwarding paths are used. The contribution of the crankback mechanism, in the interdomain framework where the complete topology and the traffic load is not known by a single entity, is significant.

Now, we analyze the results of the simulations performed with the topology containing 20 transit ASs and 10 Gbps links. In these simulations, there are 13% of congested links with CSPF, and only 2% with both “vivaldi” and “nearest NH” heuristics. This difference is due to the difference in the amount of traffic carried with each technique. We observe from figure 7.3 (b) that the amount of traffic carried inside the topology is higher with CSPF than with our heuristics. There are 19% (18%), from the total amount of LSPs, of additional LSPs established with CSPF compared to the use of “nearest NH” (“vivaldi”, respectively). Moreover, there is a difference of 32%, from the total number of LSPs, of established LSPs between CSPF and IP forwarding. The second reason for having more congested links with CSPF performed by the global PCE than with the two heuristics is that CSPF is an exact solution. It always favors short delay links over links with longer delays. This is not always the case with the heuristics because they rely on an estimation of the delay. As a consequence, short delay links become congested faster with CSPF performed by a global PCE than with the two heuristics.

Figure 7.3 (a) shows the cumulative distribution of the end-to-end delay for the LSPs established on the topology with the different path selection techniques. This distribution is almost the same for the two heuristics coupled with ERO expansion. However, we see that there are more paths with an end-to-end delay below 1927 units, with our two heuristics than with CSPF. We assume that this is due to the higher number of LSPs established with CSPF than with the two heuristics. Some LSPs with a delay shorter than 1927 units are established at the end of the simulation with our heuristics. However, with CSPF the low delay links are already congested. A short delay path may be found by the heuristics because there is less congestion in the topology than with CSPF due to the lower number of LSPs already supported by the topology.

On this large topology, crankback plays an important role. There are 45% of the established LSPs for which crankback occurs with “vivaldi” and 54% with “nearest NH”. The maximum number of crankback for the establishment of an LSP is 199 with “vivaldi” and 283 with “nearest NH”. However, there are fewer than 6 crankbacks for 90% of the LSPs established with “vivaldi” and fewer than 8 crankbacks, respectively, with “nearest NH”.



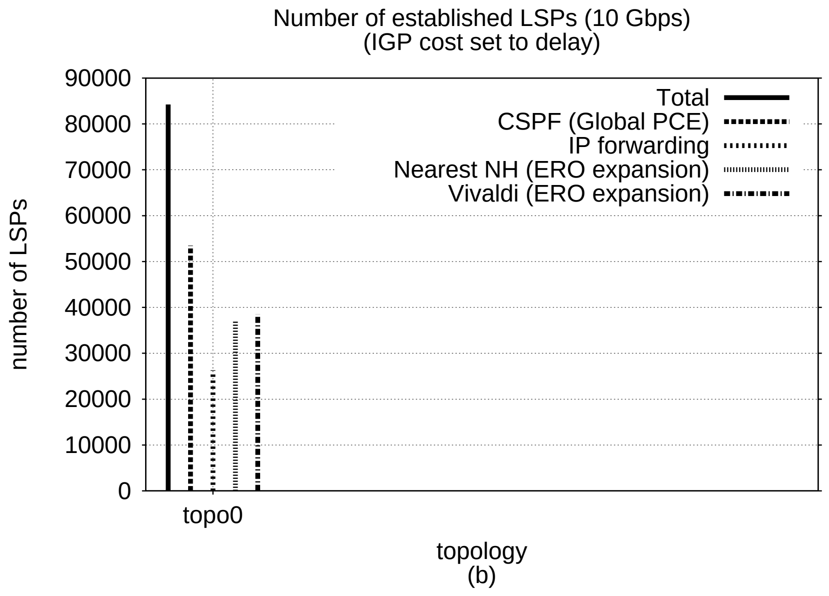


Figure 7.3: Delay of LSPs established on topology with 20 transit ASs

We have seen in chapter 6 that the maximum amount of crankback is larger when no bandwidth reservations are associated with the LSPs. There are a maximum of 293 crankbacks with vivaldi and 373 crankbacks with nearest NH, for LSPs that can be established. Moreover, there is a larger difference in the amount of crankback required to determine that a suitable delay constrained LSP cannot be established than to determine that a path with a certain delay and bandwidth cannot be found. For both established and failed LSPs, more NHs are suitable with a single than with two constraints. Thus, the establishment of an LSP subject to a single constraint is tried toward a larger amount of NHs. Each attempt can lead to crankback if the path cannot be completed at the NH or downstream of the NH. With an additional constraint, fewer possibilities to complete a path are available at each ingress ASBR. Thus, for LSPs that cannot be established, fewer paths are explored and the amount of crankback is lower. This is also true for LSPs that can be established but to a smaller extent. Moreover, there are fewer chances to be able to find a suitable path when constraints are added or more stringent constraints are used.

7.1.2 Cooperative PCEs

In this section, we analyze the establishment of delay constrained and bandwidth guaranteed LSPs with cooperative PCEs path computation. We describe the results obtained on the large topology composed on 20 transit ASs. There is more congestion on this topology. Thus, the effect of congestion is more visible.

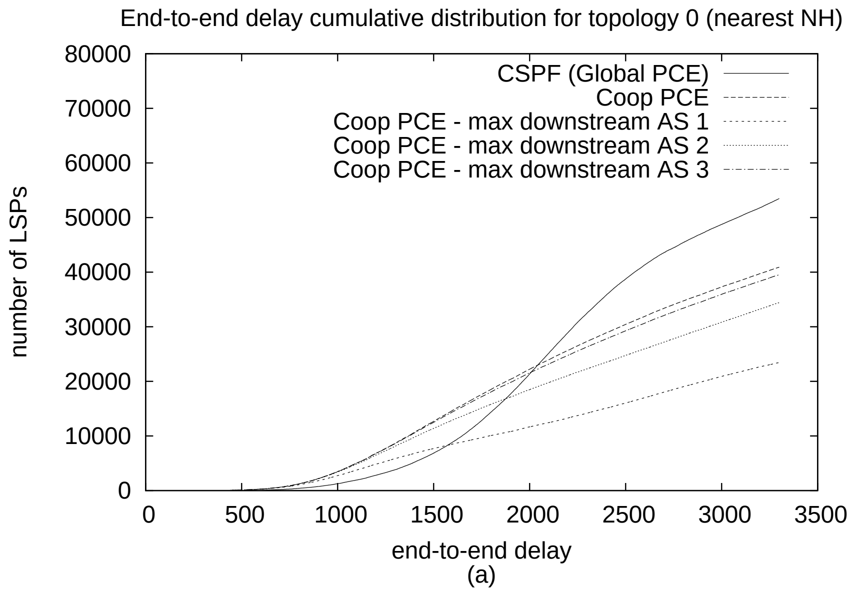
We compare the results obtained with a complete exploration of the downstream PCEs to the results achieved with a limitation of 1, 2, 3 and 4 downstream PCEs contacted by each PCE that participates in the computation. In addition, we compare these configurations of the cooperative PCEs computation technique to the CSPF computation inside a global PCE.

First, we look at the end-to-end delay of the established LSPs and at the number of LSPs that can be established on the topology. In chapter 6, we proposed two heuristics to rank a set of NHs. Additionally, we described a way to use these heuristics in order to select a subset of the downstream PCEs that will participate in the computation, from the set of downstream PCEs available for a destination.

Figure 7.4 (a) shows the cumulative distribution of the end-to-end delay for the LSPs that are computed with the nearest NH heuristic. Figure 7.4 (b) illustrates the amount of LSPs that are established with each configuration of the cooperative PCEs and the nearest NH heuristic.

Our first observation on figure 7.4 concerns the delay of the paths computed by the global PCE compared to cooperative PCEs. As in section 7.1.1, we observe that there are not many LSPs with a very low delay that result from the global CSPF computation. A complete exploration of downstream PCEs with cooperative PCEs enables to find more paths with an end-to-end delay below 1921 units than the global PCE. Again the reason for this is that short delay links become faster congested with CSPF performed on the complete topology than when the computation is distributed among elements that have a limited view of the complete topology. We observe that the limit under which there are more lower delay paths with ERO expansion than with CSPF is slightly bigger, 1927 units, as seen in section 7.1.1. However, the difference is not significant. This difference is due to the fact that ERO expansion does not aim at finding the shortest delay path but only a path that respects the delay constraint.

Second, we observe in figure 7.4 (a) that the number of LSPs with an end-to-



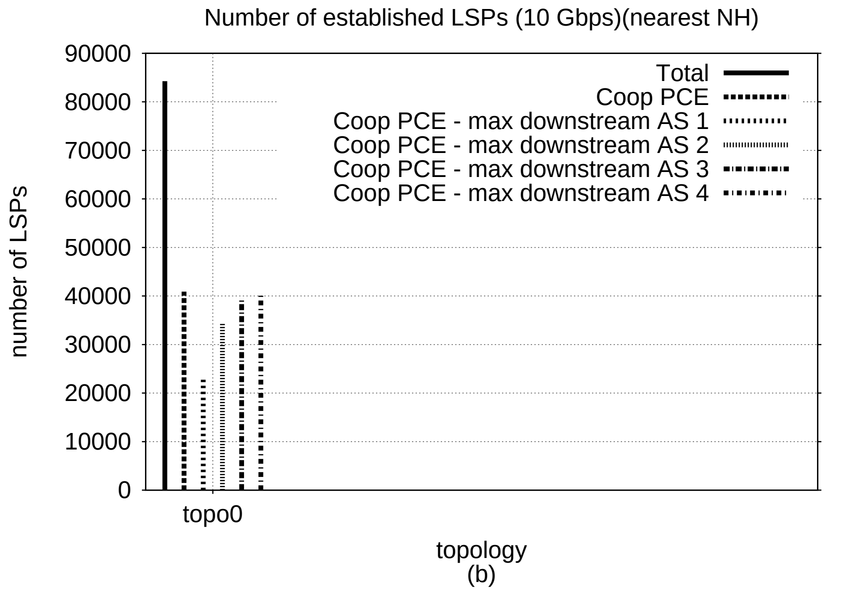


Figure 7.4: Cooperative PCEs with nearest NH on topology with 20 transit ASs

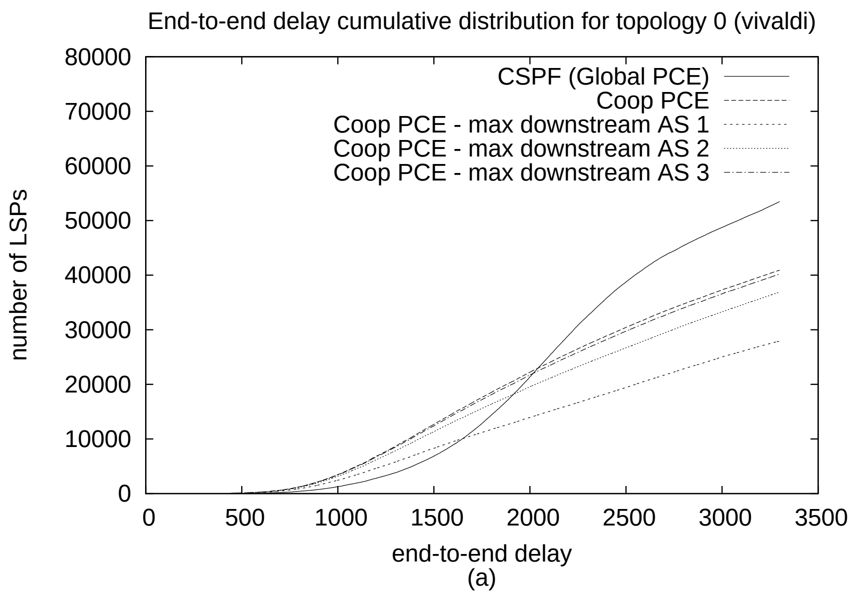
end delay below a given value increases when the limit of the number of contacted downstream PCEs increases. Moreover, the end-to-end delay cumulative distribution for the paths computed with a limit of 3 downstream PCEs is almost the same as the distribution with a complete exploration of the downstream PCEs. As in the simulations without bandwidth reservations, we observe that a limit of two downstream PCEs presents a significant improvement compared to a limit of one downstream PCE in the computation of delay and bandwidth constrained LSPs.

In figure 7.4 (b), we have the number of LSPs that can be established by the different configurations of cooperative PCEs with the nearest NH heuristic. There are 49% of the total number of LSPs that are established with a complete exploration of the downstream PCEs. In decreasing order of the exploration limit from 4 to 1, there are 48%, 47%, 41%, 28% of established LSPs. Again, we observe that there is a significant gain in the number of LSPs established with a limit of 2 compared to a limit of one downstream PCE. With the global PCE, the load of the topology is 15% higher than with a complete exploration of the downstream PCEs. 63% of the LSPs are established by the global PCE. However, 31% of the LSPs can be established along the IP forwarding paths. This is higher than an exploration of one downstream PCE with nearest NH. But, it is lower than the other configurations of cooperative PCEs.

Figure 7.5 concerns the vivaldi heuristic. It presents the same information as figure 7.4. We can make the same observations for the vivaldi heuristic than for nearest NH. That is, a limit on two downstream PCEs contacted by each PCE provides a significant improvement on the number of established LSPs compared to a limit of 1. Moreover, the end-to-end delay cumulative distribution and the number of LSPs established with a complete exploration can be approximated by cooperative PCEs with a limit of 3 downstream PCEs.

When comparing the nearest NH and vivaldi heuristics, we note that there are more LSPs established with vivaldi than with nearest NH for each limitation on the number of downstream PCEs. There are more LSPs established when the single downstream PCE is selected based on vivaldi, 33%, than nearest NH, 28%. With vivaldi and a limit of 1, more suitable paths are found than with IP forwarding. With vivaldi, already 44% of LSPs are established with a limit of 2, 48% with a limit of 3 and 4 downstream PCEs.

The upper and lower bounds on the number of messages exchanged between PCCs and PCEs during cooperative PCEs path computation are provided in figures 7.6 and 7.7. These figures concern the LSPs that can be established on the topology. For both nearest NH and vivaldi, we observe that there are more LSPs requiring a very low number of messages when bandwidth has to be reserved in addition to the delay constraint. The reason is the same as for the lower amount of crankback with ERO expansion. Due to the additional constraint, additional NHs are not suitable for the establishment of the LSP. Thus, fewer downstream PCEs are contacted, in our implementation. We note that this is not the case for the solution proposed in [VZB06] because in [VZB06] the downstream PCEs are contacted before computing the local segments. Thus, in [VZB06], all the downstream PCEs are contacted. On the upper bound curve (figure 7.6), there are 55% (i.e 22691 LSPs) of the established LSPs that require at most 50 messages with a complete exploration when bandwidth guarantees are requested. Without bandwidth reservations, only 5% of the established LSPs (i.e. 4144 LSPs) require at most 50 messages, with a complete exploration (figure 6.17). When considering the lower bound on the number



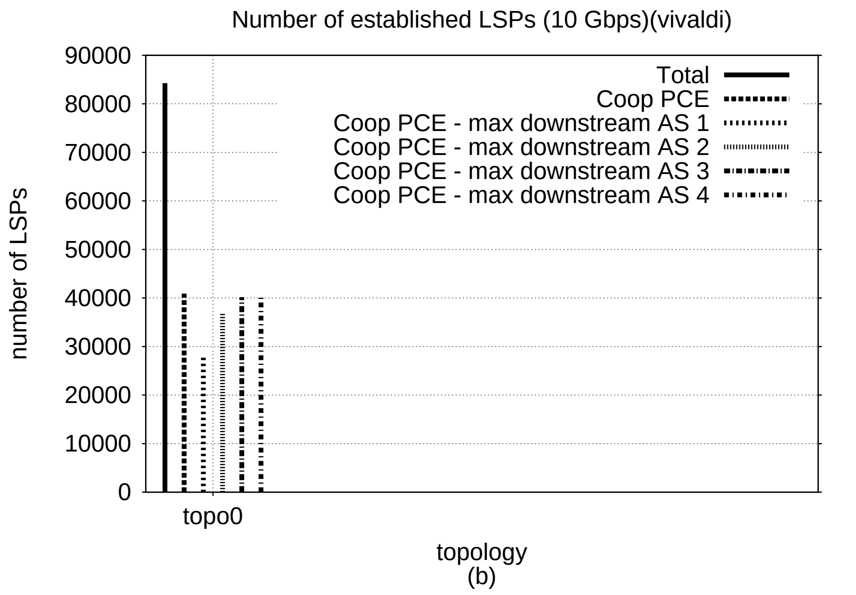
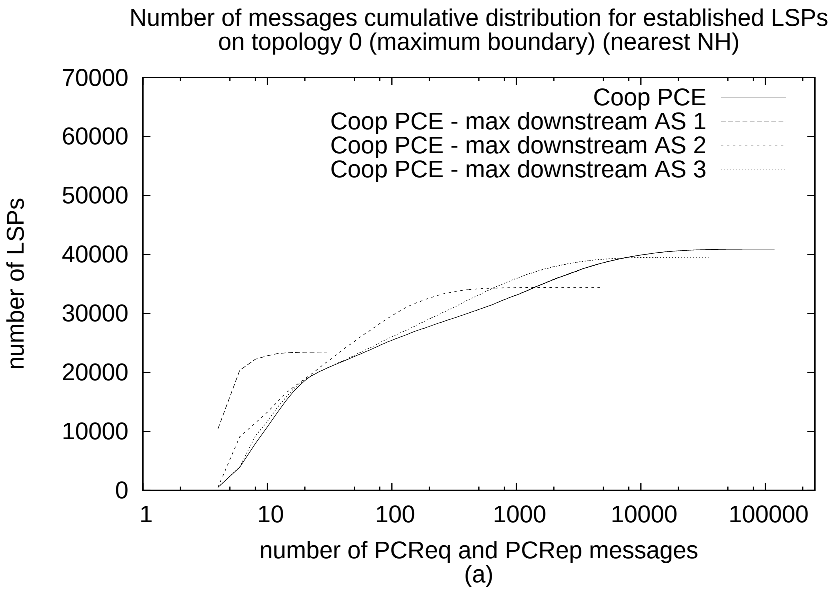


Figure 7.5: Cooperative PCEs with vivaldi NH on topology with 20 transit ASs

of messages exchanged, there are 25368 and 6441 established LSPs that may require less than 50 messages for the computation of their path, with (figure 7.7) and without bandwidth guarantees (figure 6.18), respectively.

We have seen in chapter 6 and in this section that there are IP forwarding paths with a low delay in our simulations. We noted that this is due to the use of the



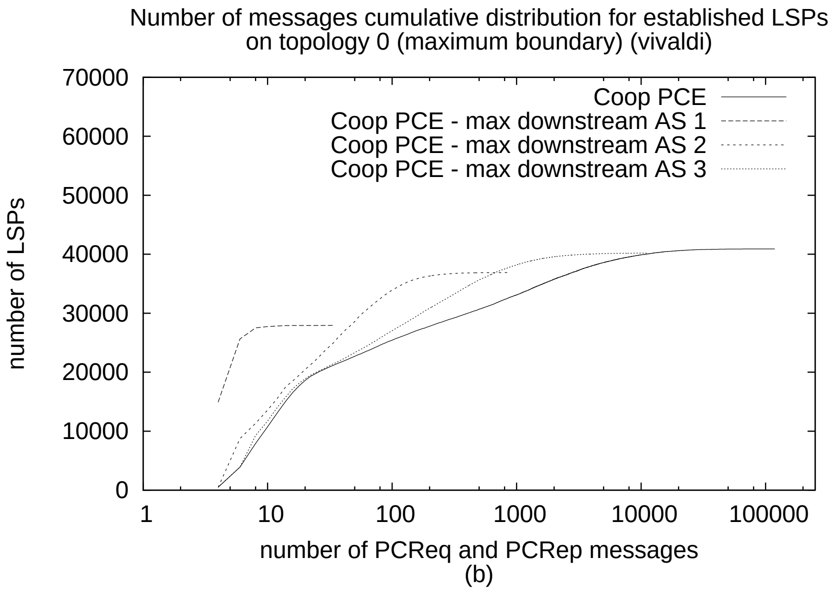
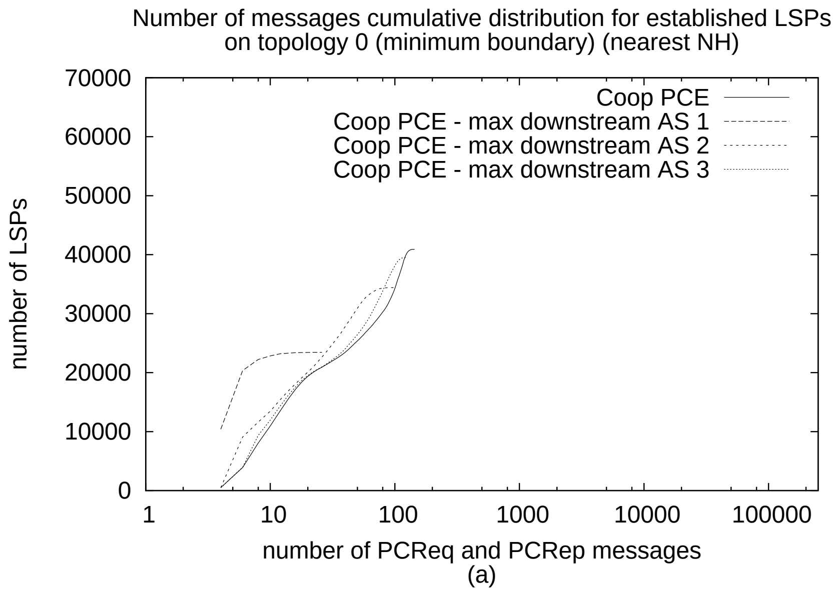


Figure 7.6: Signaling overhead with cooperative PCEs and the heuristics on topology with 20 transit ASs (upper bound)

propagation delay as the IGP cost of the links. In such a setting, if BGP selects a route based on the IGP cost, BGP prefers routes with a short delay to the NH. We performed other simulations with a unitary IGP cost of the links. These simulations show that less LSPs, with delay constraints and bandwidth reservations, can be



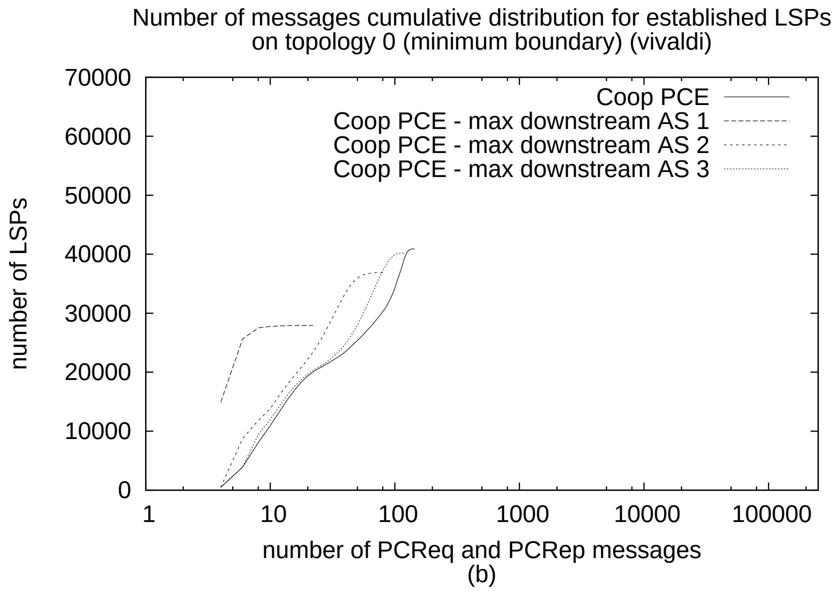


Figure 7.7: Signaling overhead with cooperative PCEs and the heuristics on topology with 20 transit ASs (lower bound)

established along IP forwarding paths than when the delay is used as IGP cost. Moreover, in these simulations, the difference in the results between the nearest NH and vivaldi heuristics is slightly larger than in the simulations presented in this document. More LSPs can be established with vivaldi than nearest NH for the ERO expansion technique. Moreover, more LSPs are established with cooperative PCEs and a limit of one downstream PCE contacted based on vivaldi than based on the nearest NH heuristic.

In this section, we have evaluated the two heuristics proposed in chapter 6, for the computation of delay and bandwidth constrained paths. In our analysis, we have considered the ERO expansion and cooperative PCEs techniques. For both techniques, we have shown that the vivaldi heuristic performs better than the nearest NH heuristic. Compared to nearest NH, more LSPs can be established with vivaldi. With vivaldi there are more paths with a small delay than with nearest NH. Less crankback is required in the ERO expansion technique when vivaldi is used instead of nearest NH. Finally, fewer messages are exchanged between cooperative PCEs with vivaldi than with nearest NH.

We have also shown that the performance of IP forwarding degrades when congestion occurs. Once a link is congested, LSPs cannot be established toward any of the destinations that are reached via this link. Such a congested link cannot be avoided for subsequent LSP establishments as it is the case with RSVP-TE.

Third, we observed that the global PCE performing CSPF computation is not able to find many paths with a very short delay in a congested network. Links with small delays are used in the first establishments of LSPs. They quickly become congested. Once they are congested they cannot be used for other LSPs.

Finally, we noted that the amount of crankback and messages exchanged for the computation of delay and bandwidth constrained LSPs is lower than for the computation of LSPs that are only subject to the delay constraint. This comes from the fact that fewer path alternatives are available when the constraints are stricter. Thus, LSPs with stricter constraints also have fewer chances to be established as we have seen in our simulations.

Table 7.1 gives an overview of the properties observed for the different path computation techniques in the simulations of section 7.1.1 and this section, on congested topologies. Compared to table 6.1, we observe that the global PCE does not provide many short delay paths when bandwidth reservations are required. With IP forwarding, it is not possible to establish many LSPs in a congested network. Finally, as mentioned earlier, cooperative PCEs and ERO expansion require the exchange of less messages between PCCs and PCEs, in average. However, there are still LSPs that require the exchange of many messages with a complete exploration by cooperative PCEs.

7.2 Interactions with Intra-domain TE Algorithms

We have seen earlier that one of the objectives of traffic engineering is to provide QoS to the users. In addition, TE aims at distributing the traffic inside a network in order to reduce the cost of the network by delaying upgrades and to improve revenue by allowing to support as much traffic as possible in the network, to satisfy as many users as possible.

	Criteria			
	Estab LSPs	Low delay LSPs	Crankback	PCEP msg
Global PCE	very good	bad	–	excellent
IP forwarding	bad	fair	–	–
ERO expansion	good	good	good	very good
Cooperative PCEs (complete exploration)	good	good	–	fair
Cooperative PCEs - maxdownstream 1	bad	fair	–	very good
Cooperative PCEs - maxdownstream 2	good	good	–	good

Table 7.1: Overview of the simulations results for delay constrained LSPs with bandwidth reservations

Up to now, we mostly considered the provision of QoS of this thesis. We did not focus on the distribution of the traffic on the topology. However, we already took into account the amount of LSP requests that could be satisfied, in our evaluation of different path computation techniques. The purpose of this section is to study the impact of intra-domain Traffic Engineering techniques on the QoS that can be provided across domain boundaries.

In practice, the different domains may use different techniques to engineer their traffic. Each domain is allowed to optimize any metric that it wants, independently of the optimization performed by its neighboring domains. An operator may want to reduce the cost of the traffic inside its network, to reduce the average link loads, to reduce the end-to-end delay, . . . Certain of these objective may go against the TE objectives implemented in other domains. Moreover, they may be antagonist to the users' QoS requirements.

In this section, we show, by means of simulations, that certain intra-domain TE objectives may cohabitate without problems with certain types of inter-AS QoS requirements. Moreover, we show that while certain objectives are opposed to certain inter-AS QoS requirements, it is possible to achieve a suitable trade-off between the fulfillment of these intra-AS TE objectives and inter-AS QoS requirements. To demonstrate this, we assume that the TE technique used inside all the domains is the same. It is called DAMOTE (Decentralized Agent for MPLS Online Traffic Engineering). This technique has been proposed in [BML03b]. This algorithm aims at distributing the load on the links and reducing the total amount of traffic carried on the links of an intradomain topology. It is available in the TOTEM toolbox (TOolbox for Traffic Engineering Methods) [BCED$^+$]. The objective function that is minimized by DAMOTE, in our simulations, is provided in equation 7.1.

$$loadBal * \sum_{(i,j)\in\mathcal{U}} \left(\frac{L_{(i,j)}}{w^{cap}(i,j)} - \frac{\bar{L}}{w^{cap}} \right)^2 + tMin * \sum_{(i,j)\in\mathcal{U}} \left(\frac{L_{(i,j)}}{w^{cap}(i,j)} \right)^2 \quad (7.1)$$

$$with \frac{\bar{L}}{w^{cap}} = \frac{1}{|\mathcal{U}|} \sum_{(i,j)\in\mathcal{U}} \frac{L_{(i,j)}}{w^{cap}(i,j)} \quad (7.2)$$

where $\mathcal{U}$ is the set of edges in the topology, $w^{cap}(i,j)$ is the capacity of link (i,j), $L_{(i,j)}$ is the load of link (i,j). By default, DAMOTE is configured with loadBal equals to 2 and tMin equals to 1. We used these default values.

Equation 7.2 is the mean of the relative link loads throughout the network. Thus, the first term of the objective function 7.1 is the variance on the relative link load. It represents the deviation from the optimal load balancing situation. In a perfectly balanced network, this deviation would be zero for all links so that all links would be used in exactly the same proportions. The objective of this term is to flatten the relative load throughout the network. If this term was used alone long paths could be returned in order to achieve a good load-balancing solution. However, if paths are long, bandwidth consumption inside the network is high. Thus, the second term aims at minimizing the path length of the LSPs. In equation 7.1 the terms are weighted by factors, $loadBal$ and $tMin$, that can be passed as parameters to the tool. The (weighted) combination of both terms will give more importance to the load-balancing term if the deviation is high enough to justify the detour, else it will let the "shortest path" term minimize the resources used [BCED$^+$].

7.2.1 Interactions between DAMOTE and our Simulator

In our simulations, the DAMOTE algorithm of the TOTEM toolbox provides the intradomain path segments. Now, a PCE, that receives a request for a path, computes the intradomain paths segments with the DAMOTE algorithm instead of CSPF. The PCE computes paths toward each NH available for the destination, based on the BGP routes. The delay of the path segments provided with DAMOTE is used in the ranking of the NHs with nearest NH and vivaldi. We note that with vivaldi the delay estimation of the path from the NH to the destination is also used in the ranking.

The path segments toward each NH are computed with DAMOTE independently of one another. That is, bandwidth is not reserved along these path segments before a choice is made to use one of them for the LSP. Then, bandwidth is only reserved along the chosen path segment. This avoids that the path segment to one NH varies depending on the path segments to other NHs that are computed before this path segments. The computation of the segments depends only on the paths of the LSPs that have already been established on the topology.

7.2.2 Simulations with DAMOTE

The simulations with the DAMOTE algorithm are run on a topology of 10 transit ASs. This topology is generated with the method presented in section 6.3.1 for the topologies composed of 5 transit ASs. The ratio of the total bandwidth in the network over the amount of traffic (T) to be carried on the topology, $\sum_{(i,j)\in\mathcal{U}} \frac{w^{cap}(i,j)}{T}$, is 40.72 for topology 0 that is composed of 5 transit ASs. This ratio is much lower and equal to 13.08, for the topology with 10 transit ASs that is considered in this section. This is similar to this ratio for the topology composed of 20 transit ASs, 14.43. We use a topology composed of 10 transit ASs because it has a heavier traffic demand to support compared to the capacity of the network than the topologies composed of 5 transit ASs. Thus, it is more appropriate than the small topologies to measure the effect of load-balancing mechanisms. Moreover, the topology considered in this section has a similar ratio as the topology composed of 20 transit ASs but it is smaller. Thus, the simulation time for the 10 transit ASs topology is reduced compared to the large topology. The bandwidth of the links is set to 2400 Mbps. A total of 10585 LSPs are to be established on the topology. These LSPs have a bandwidth guarantee of 100 Mbps. We perform two sets of simulations. First, there is no delay constraint for the LSPs. Second, a maximum end-to-end delay of 2700 units has to be respected by the LSPs.

For each set of simulations we compare the use of DAMOTE inside transit ASs to a simple CSPF inside the ASs. The interdomain path computation technique that is considered in this section is ERO expansion.

First, we performed simulations without an end-to-end delay constraint associated with the LSPs to look at the amount of traffic supported with intradomain load-balancing. The results of these simulations are shown in figures 7.8 and 7.9. In figure 7.8, CSPF is used to compute the path segments inside the ASs. The results with DAMOTE are shown in figure 7.9.

The x-axis in each figure presenting the end-to-end delay of the paths ends at 2700 units in figures 7.8 and 7.9. This value is the delay constraint that will be used in the next simulations. However, in the simulations without delay constraint, LSPs with a longer delay are established. The total number of LSPs established by each path computation technique is shown in figures 7.8 (b) and 7.9 (b).

We observe from figure 7.8 that there are more LSPs with a delay below 2700 units with ERO expansion (vivaldi and nearest NH) than with the global PCE when the intradomain path segments are computed with CSPF. With DAMOTE (figure 7.9), we note that the number of LSPs with a delay below 2700 units is almost the same with ERO expansion and the global PCE. This is good for an algorithm that aims at distributing the load inside ASs without taking propagation delay into account. However, we note that the maximum end-to-end delay of the paths computed with DAMOTE is much higher than with CSPF. With DAMOTE and vivaldi, the path with the highest delay has a delay of 21849 units. With CSPF and vivaldi, the maximum delay among the established LSPs has a delay of 6921 units.

When looking at the total number of LSPs without delay constraint established

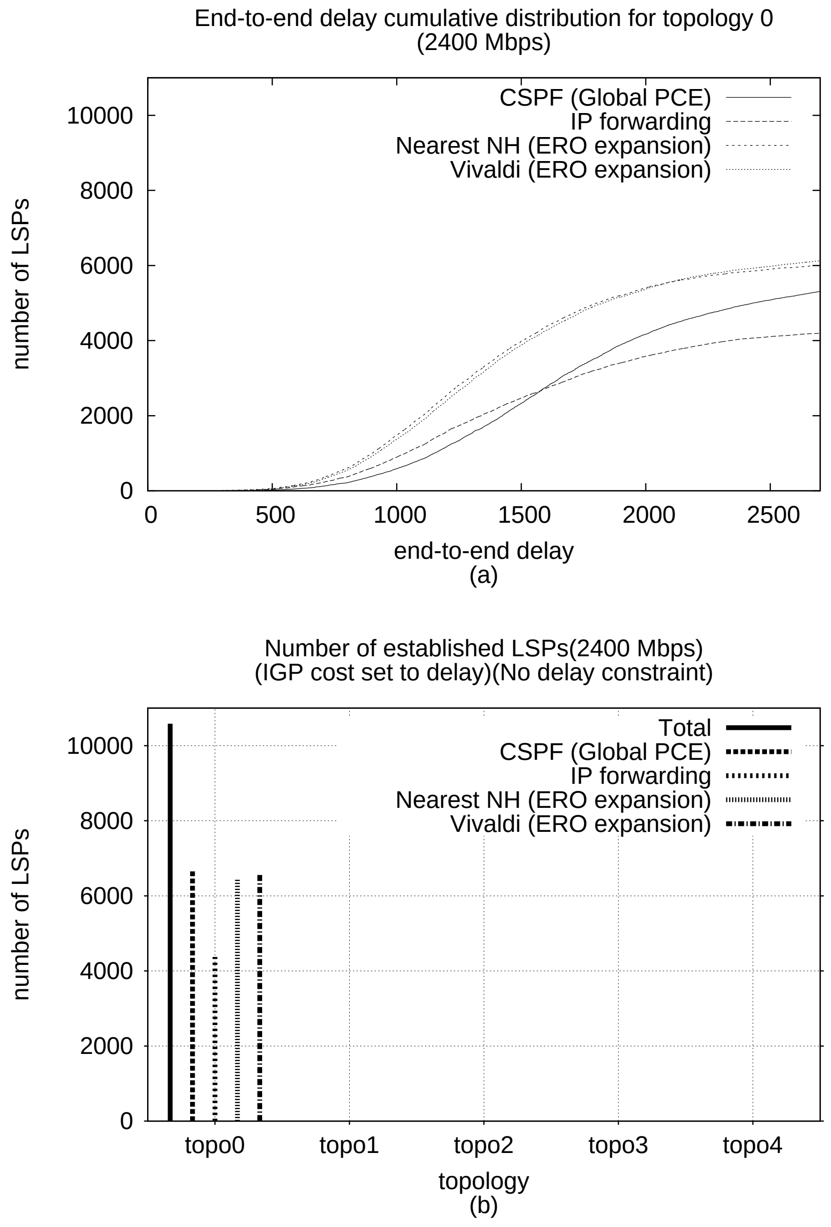
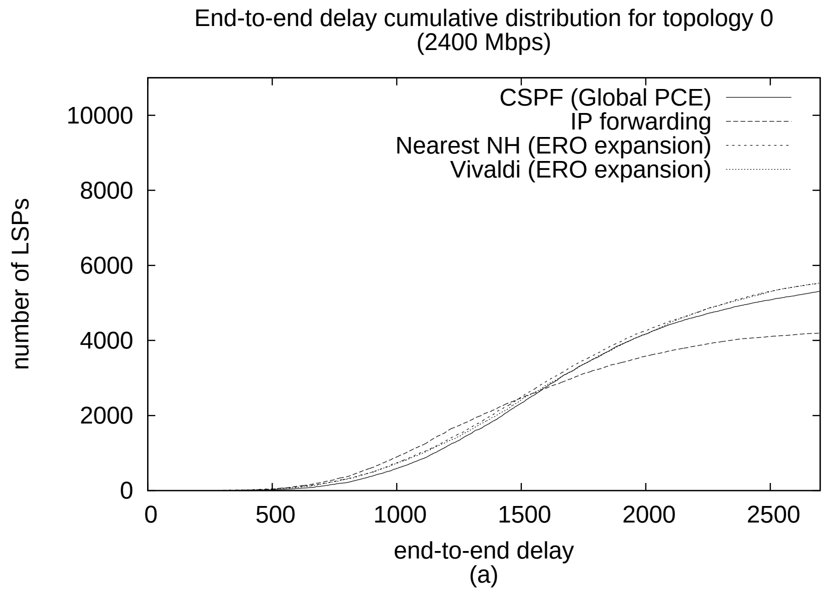


Figure 7.8: LSPs established with CSPF inside an AS on topology with 10 transit ASs without delay constraints

by each computation technique, we observe that there are 63% out of the total LSP's demand that can be established by the global PCE (figures 7.8(b) and 7.9 (b)). There are 62% of established LSPs with ERO expansion and the vivaldi heuristic, independently of the use of DAMOTE or CSPF for the computation of



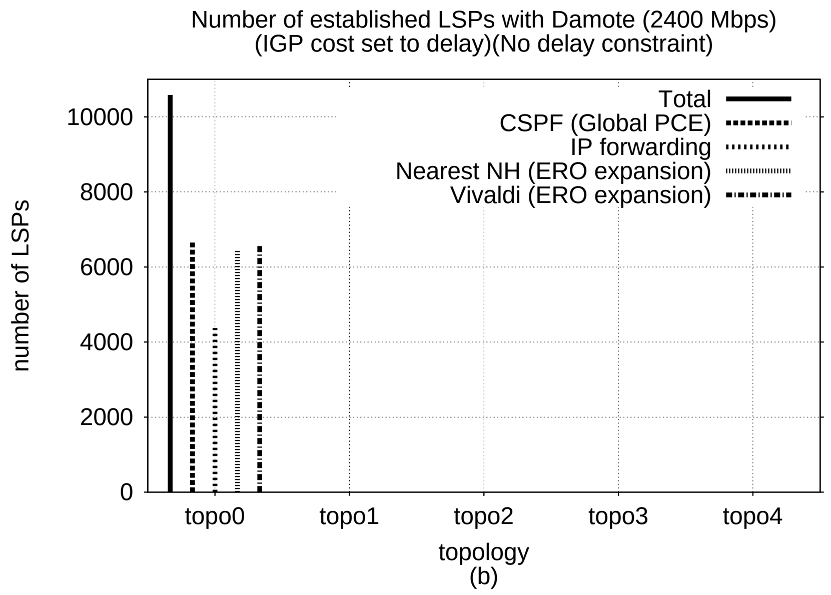
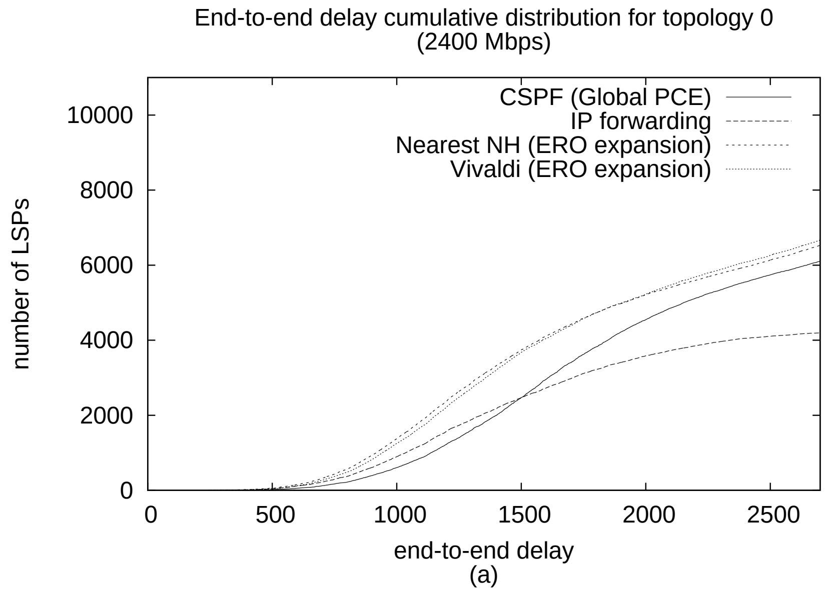


Figure 7.9: LSPs established with DAMOTE on topology with 10 transit ASs without delay constraints

intra-domain path segments. With nearest NH, 61% of the LSPs are established with CSPF intra-domain path computation and 60% with DAMOTE intra-domain TE technique.

Figure 7.10 and 7.11 illustrate the number of delay and bandwidth constrained

LSPs that are established on the topology with 10 transit ASs. CSPF is used inside the ASs to obtain the results of figure 7.10. Figure 7.11 is generated from the simulation with DAMOTE as intra-domain TE technique.



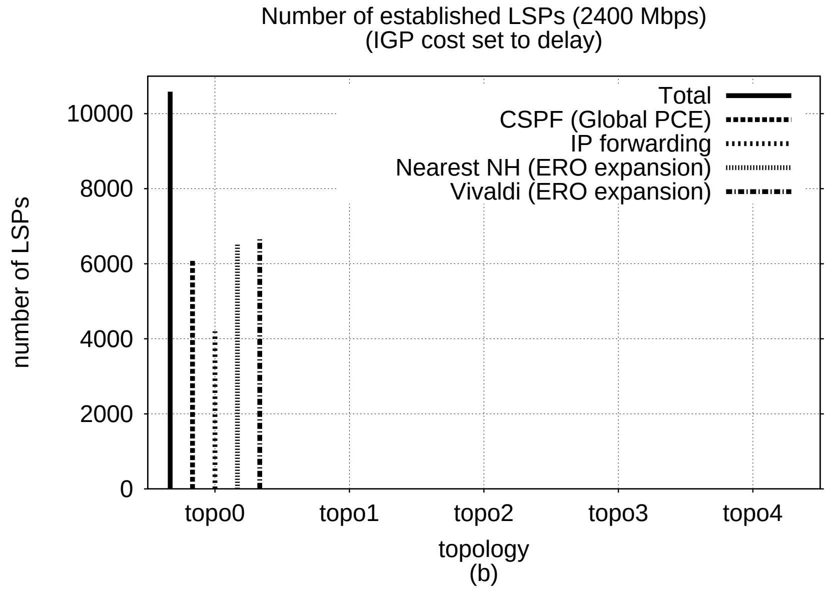
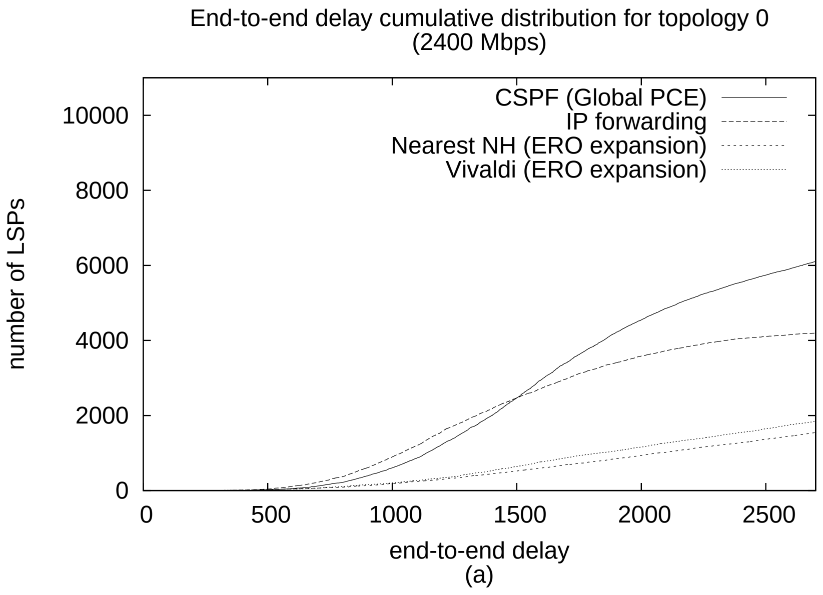


Figure 7.10: LSPs established with CSPF inside an AS on topology with 10 transit ASs

In figure 7.10, we observe the familiar form of the curve that occurs in a con-



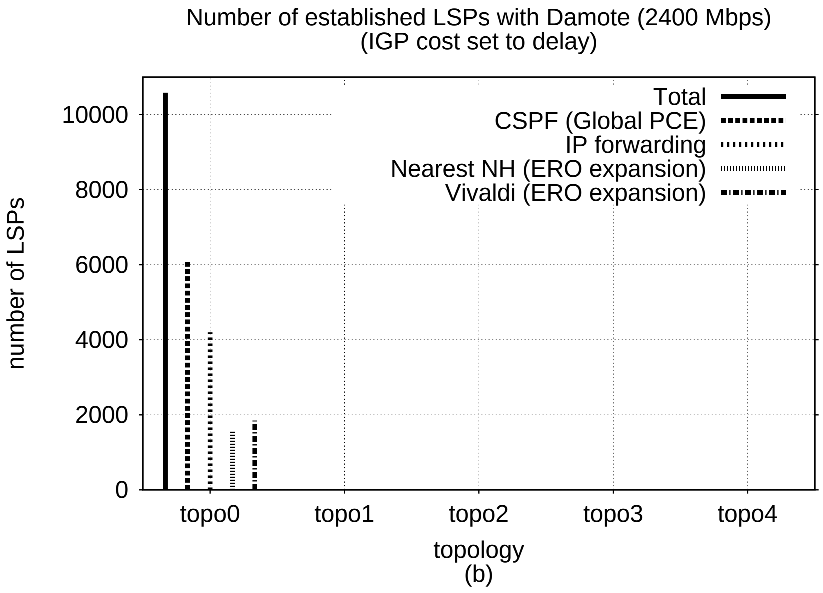


Figure 7.11: LSPs established with DAMOTE on topology with 10 transit ASs

gested network with CSPF computation inside a global PCE. With a global PCE there are 6109 out of the 10585 LSPs (58%) that can be established. There are 63% of established LSPs with vivaldi, 62% with nearest NH and 40% with IP forwarding. We see in figure 7.11 that the percentages of LSPs established with vivaldi and nearest NH drop to 17% and 15%, respectively, with DAMOTE used inside the transit ASs. Since DAMOTE is not used in the computation by the global PCE and for the computation of the IP forwarding paths, the results for these two techniques are unchanged.

From figures 7.10 and 7.11, we note that DAMOTE has an impact on the delay of the computed path segments and, thus, on the delay of the end-to-end paths that can be computed. Because DAMOTE balances the traffic on the intradomain topologies without considering the propagation delay of the paths, the paths have a higher delay. Consequently, many LSPs cannot be established due to their maximum end-to-end delay constraint.

Surprisingly, when no delay constraint is associated with the LSPs, we have seen that more LSPs with a delay below 2700 units can be established. The reason for this is that DAMOTE provides the same set of paths from an ingress ASBR to a set of NHs until one of these paths is used. If these paths have a high delay, multiple subsequent LSP establishments may be rejected while if an LSP is established along one of these paths segments other links get preferred by DAMOTE for the next LSPs. These links may have a lower delay and thus paths using these links may have a low delay. With a maximum end-to-end delay constraint, LSP establishment may be blocked inside an AS because the links that are preferred have a high delay. However, establishing a few LSPs through the AS may enable to find paths with a low delay through the AS. Thus, DAMOTE is suitable for the provision of bandwidth guaranteed LSPs. Moreover, a good fraction of the computed paths have a low end-to-end delay, even if DAMOTE does not take propagation delay into account.

Table 7.2 shows the average link loads for the simulations presented in this section. The average links load, between the CSPF and DAMOTE intra-domain path computation techniques are not comparable, for the simulations with delay constraints. With delay constraint the amount of traffic supported with CSPF intra-domain path computation is much higher than with the DAMOTE algorithm. Thus, it is normal that the average link loads are higher with CSPF than with DAMOTE.

	Delay constraint		No delay constraint	
	CSPF	DAMOTE	CSPF	DAMOTE
Global PCE	600 Mbps	–	716 Mbps	–
Vivaldi	559 Mbps	157 Mbps	557 Mbps	643 Mbps
Nearest NH	547 Mbps	133 Mbps	542 Mbps	626 Mbps

Table 7.2: Average link load

For the simulations without delay constraints, we note, in table 7.2 that the average link loads are higher with DAMOTE than with CSPF for the computation of intra-domain path segments. This is true for both the vivaldi and the nearest NH heuristics. However, here the total amount of traffic supported on the topology is similar between CSPF and DAMOTE. The average link loads are higher with DAMOTE because there are paths crossing a higher number of links with DAMOTE compared to CSPF.

When we look at the percentage of congested links, that is the percentage of links that cannot support an additional LSP, in table 7.3, we see that there are less congested links with DAMOTE than with CSPF for LSPs without delay constraints and comparable amount of traffic supported. This illustrates the power of DAMOTE in distributing the traffic on the topology in order to avoid congestion.

	Delay constraint		No delay constraint	
	CSPF	DAMOTE	CSPF	DAMOTE
Global PCE	10.76%	–	15.56%	–
Vivaldi	4.80%	0.03%	4.28%	2.60%
Nearest NH	4.68%	0.05%	4.00%	2.54%

Table 7.3: Percentage of congested links

We also performed simulations with other parameters for the DAMOTE algorithm. Instead of the default parameters, we used a value of 1 for $loadBal$ and 3 for $tMin$ in the objective function 7.1. These parameters give a higher weight to the total load of the network compared to the distribution of the traffic in the network, in the objective function 7.1. This configuration should favor the computation of paths with a small number of hops inside ASs over a good balance of the traffic inside ASs.

In these simulations, we observed that the number of delay constrained LSPs established with DAMOTE inside the ASs is almost the same as with CSPF. Thus, these parameters are suitable for the provision of paths with maximum end-to-end delay constraints and minimum bandwidth reservations. Moreover, in the simulations without the delay constraint, more established LSPs have a lower delay than the delay constraint, used in the simulations with LSPs subject to a maximum end-to-end delay, with the new parameters than with the default parameters. We also noted a much lower maximum delay in the simulations without the delay constraint. With the default parameters, there was one LSP whose delay was equal to 21849 units with the vivaldi heuristic. With the new parameter, the maximum delay observed is 9379 units.

The new parameters enable to favor shorter over longer intra-AS paths. This is verified by our simulations. When there is no delay constraint associated to the LSPs, the maximum hop count observed with the new parameters is much lower than with the default parameters. It is equal to 23 in the first case and to 66 in the latter case, with the vivaldi heuristic. The average hop count is the same with the new parameters for DAMOTE and CSPF. This has an impact on the average link load. With the new parameters the average link loads are similar to the average link loads in the simulations with CSPF used inside the ASs.

Finally, for a comparable number of LSPs established there is an improvement in the number of congested links with the new parameters for the DAMOTE algorithm than with CSPF. The new parameters enable to support a similar number of delay constrained LSPs with DAMOTE and CSPF, inside ASs, with a lower percentage of congested links. As we see in table 7.3, there are, for delay constrained LSPs, 4.8% of congested links with CSPF and the vivaldi heuristic. With the new parameters for DAMOTE, there are 2.72% of congested links,

In table 7.4, we show a qualitative appreciation of each intra-AS computation technique in the computation of inter-AS constrained LSPs, according to different criteria. We see in the table that DAMOTE, with its default parameter values, is not appropriate for providing strict maximum end-to-end delay guarantees. Only a few LSPs can be established. Hence, there is a bad appreciation for both the number of established LSPs and the number of low delay LSPs. The good appreciation of DAMOTE, with the default parameters, for link loads and congested links in the simulations with delay constrained LSPs is due to the very low number of LSPs that are established. However, the results achieved without maximum end-to-end delay constraints are good, with the default parameters for DAMOTE. Without a strict delay constraint, many LSPs have a low end-to-end delay. Moreover, the percentage of congested links is lower with DAMOTE (default parameters) than with the CSPF algorithm inside the ASs.

	Delay constraint		
	CSPF	DAMOTE (default)	DAMOTE (loadBal=1, tMin=3)
Estab. LSPs	good	very bad	good
Low delay LSPs	good	very bad	good
Hop count	good	good	good
Link load	good	very good	good
Congested links	fair	very good	good
Max delay	–	–	–
	No delay constraint		
	CSPF	DAMOTE (default)	DAMOTE (loadBal=1, tMin=3)
Estab. LSPs	good	good	good
Low delay LSPs	good	good	good
Hop count	fair	very bad	fair
Link load	good	good	good
Congested links	fair	good	good
Max delay	good	very bad	fair

Table 7.4: Overview of the simulations with DAMOTE

More importantly, we observe in table 7.4 that there are parameters for DAMOTE that enable to achieve much better results than the use of DAMOTE with the default parameters. The DAMOTE algorithm, with $loadBal$ set to 1 and $tMin$ set to 3, enables to achieve results that are as good as with a CSPF computation inside ASs. In this case, DAMOTE computes short delay intra-AS path segments because it considers the length of these segments in its optimization ($tMin = 3$). In addition, this configuration of DAMOTE enables to reduce the number of congested links in the topology compared to the use of CSPF inside ASs.

From the observations of this section, we conclude that the use of a load-balancing TE algorithm inside ASs can contribute to a reduction in the link loads and congested links inside ASs while still allowing good performance of the inter-domain path computation techniques. We predict that, even if the load-balancing algorithm does not try to minimize the length of the intra-AS paths, as when the default parameters are used with the DAMOTE algorithm, good results can still be achieved. The quality of the results with techniques that do not consider the path length will depend on the proportion of delay and bandwidth constrained LSPs versus bandwidth guaranteed LSPs in the traffic demand.

7.3 Conclusion

This chapter is divided into two parts. First, we evaluated the heuristics designed for LSPs with a maximum end-to-end delay in the computation of LSPs with a delay constraint and a bandwidth reservation. We have seen that around 20% fewer LSPs could be established with ERO expansion than with the global PCE. Moreover, we noted that a complete exploration with cooperative PCEs allowed to establish around 5% additional LSPs compared to ERO expansion. Our second observation concerned the slight improvement of the vivaldi heuristic with regard to nearest NH. A small portion of additional LSPs can be established, fewer crankback is required in ERO expansion and fewer PCC-PCE messages are exchanged between cooperative PCEs with vivaldi than with nearest NH.

In the second part of this chapter, we studied the interactions of an intradomain TE algorithm, DAMOTE, and the computation of interdomain LSPs with QoS requirements. We have noticed in our simulations that a load-balancing algorithm can perform well in the provision of LSPs that are only subject to a bandwidth reservation. In this case, we have observed and explained why short end-to-end delay paths are found. Then, we have seen that if such a TE algorithm that aims at distributing the traffic inside an AS also tries to minimize the length of the computed paths, it is appropriate for the provisioning of interdomain LSPs with delay and bandwidth constraints.

Chapter 8

Conclusion

After having considered the engineering of the traffic inside Autonomous Systems (ASs) and the provision of Quality of Service (QoS) to users connected to the same Service Provider (SP), research is now lead on engineering the interdomain traffic and provisioning services with quality requirement to users connected to different SPs. In this thesis, we have addressed the latter problem.

One of the objectives of interdomain traffic engineering is the distribution of the traffic on ASs' peering links. In this thesis, we have presented the TE techniques available today with BGP. We have shown that it is easier to control the distribution of the outgoing traffic on the peering links than to engineer the incoming traffic. Engineering the outgoing traffic implies to control the selection of the BGP routes at the routers inside the AS. However, the incoming traffic flow depends on the selection of the BGP routes at distant routers. This selection depends on the TE and the routing policies of distant ASs. We have evaluated through active measurements, implying a large number of sources, two techniques, `AS-path` prepending and `Communities` that aim at influencing the selection of the BGP routes at distant routers. In our measurements we have shown that `AS-path` prepending is too coarse. This technique does not allow a precise control on the traffic. In addition, our measurements have shown that the BGP communities allow a finer control on the incoming traffic. However, the impact of a given community on the incoming traffic cannot be predicted before its application. Moreover, there exists a large number of combinations for the communities. Thus, it is not straightforward to use the BGP communities to achieve a certain TE objective. We conclude that BGP TE techniques are not sufficient for a precise control of the interdomain traffic.

MultiProtocol Label Switching (MPLS) is already used by SPs to engineer and gather statistics on their intradomain traffic. MPLS Label Switched Paths (LSPs) can be established along a specific path with the RSVP-TE signalling protocol. In this thesis, we proposed extensions to the RSVP-TE protocol for the establishment of inter-AS TE LSPs. These extensions do not require much changes in the format of the objects signalled in RSVP messages. They can be easily supported by the routers. Moreover, only the routers at the border of the ASs need to support these changes. These extensions enable to establish inter-AS LSPs toward prefix and AS destinations for TE purposes as well as to protect inter-AS LSPs against link, node and SRLG failures. We have defined two new notions for SRLGs : SRLG and SRLG ID scopes. We have shown that these scopes define the protection mechanisms, global or local repair, that are required to provide SRLG protection.

BGP is currently the only way to obtain information concerning paths to destinations outside an AS. We have seen that this protocol does not distribute enough information for a single node to be able to compute the end-to-end path for inter-AS LSPs with QoS requirements. We have described the current state of development at the IETF of two distributed path computation techniques: ERO expansion and cooperative PCEs. We have provided a theoretical comparison of these techniques and a description of their implementation. When we implemented these techniques in our simulator, their standardization was still in infancy leaving room for interpretation and implementation choices. We have showed in appendix A that our implementation choices produce the same paths as the techniques under standardization today.

Inside an AS, there are different ways to configure iBGP sessions. In this document, we considered full-meshes of iBGP sessions and the redistribution of the routes with Route-Reflectors (RRs). We have studied the diversity of the paths that can be computed distributedly based on the BGP routes available at routers participating in an iBGP full-mesh and at RR-clients. We have shown by simulations that the ERO expansion technique is not able to provide end-to-end link and node disjoint paths based on the BGP routes available at these routers. In addition, our simulations have highlighted the fact that certain nodes, the RR, are more capable of computing diverse paths than their clients. From these observations, we recommended the distributed computation of constrained paths at PCEs with TED containing all the BGP routes learned at the routers inside the domain of the PCE. This enables PCEs to compute the best paths that are possible with BGP.

To cope with the absence of QoS information associated with the BGP routes, we proposed to rely on heuristics to determine the quality of these routes. The heuristics assign a preference to the Next-Hops (NHs) of the BGP routes. The LSP is established toward the preferred NH in the ERO expansion technique. We have proposed two heuristics that aim at estimating the delay quality of the paths through each available NH for a destination. The first heuristics, nearest NH, prefers NHs that are reachable with the smallest delay path. The second heuristic, vivaldi, relies on synthetic coordinates. For both delay and delay plus bandwidth constrained LSPs, we have seen that our heuristics provide similar results with the vivaldi heuristics being slightly better than nearest NH.

We compared the performance of ERO expansion and cooperative PCEs. We made the following observation for delay constrained LSPs. First, the paths obtained with cooperative PCEs have lower delays than the paths computed with the ERO expansion heuristics. The former paths are the lowest delay paths that can be achieved based on the BGP routes, without bandwidth reservations. Second, we said that cooperative PCEs either required to maintain states inside PCEs or exchange a lot of messages in order to compute these low delay paths. Fewer messages are exchanged for the computation of most constrained LSPs with the ERO expansion heuristics when a path respecting the constraint can be found than with cooperative PCEs. Moreover, the paths computed by these heuristics are not far from the best delays available to a computation method that relies on the BGP routes. We have seen that these two observations are also true for the computation of maximum delay and minimum bandwidth guaranteed LSPs.

To reduce the number of messages exchanged by cooperative PCEs, we proposed to use our heuristics to reduce the set of downstream PCEs that are contacted by each PCE to a maximum number of PCEs. For the topologies used in our simulations, we observed that contacting 4 downstream PCEs was a good approximation to a complete exploration. We noted that contacting two downstream PCEs was a good compromise in the number of LSPs that can be established and the signalling overhead.

Service Providers may want to engineer their intradomain traffic in addition to providing QoS for interdomain traffic. We studied the interactions of an intradomain TE algorithm, DAMOTE, and the computation of interdomain LSPs with QoS requirements. This TE algorithm aims at balancing the traffic inside an AS. We observed in our simulations that such an algorithm can perform well to provision interdomain LSPs that are only subject to a bandwidth reservation. In this case, we have observed and explained why short end-to-end delay paths are found. We have also noted that such an algorithm is suitable for the establishment of interdomain LSPs subject to delay constraints if it tries to minimize the path length in addition to balance the traffic.

8.1 Overview of the Contributions

This thesis brings the following contributions to the current literature on interdomain Traffic Engineering with MPLS and BGP:

Active measurements of BGP TE techniques: These were the first BGP TE measurements that were performed. In our measurements, BGP routes are injected in the Internet and, contrary to other subsequent studies, the impact on a large number of sources is actively measured.

Extensions to RSVP-TE for inter-AS LSP establishment and protection:
[PB02] was the first publication on the subject and guided much of the actual work in this area.

A study of the impact of BGP routes on the computation of inter-AS constrained paths: This study led to recommendations on the location of the PCEs inside an AS.

The elaboration of distributed path computation techniques: Modifications, implementation and evaluation of ERO expansion and cooperative PCEs techniques. This work has been done in parallel to the development of solutions that are still under standardization at the IETF.

8.2 Further Work

Research on interdomain end-to-end QoS is still in its infancy. The two techniques, [VAZ06] and [VZB06], on which the techniques studied in the thesis are based, are still in the process of standardization. The first technique is in the final steps while the latter became Working Group documents in July 2006. Among the other proposals for interdomain end-to-end QoS there is the Mescal approach [HFP$^+$05] that proposes a way to negotiate QoS between ASs that do not belong to the same organization. [FBR$^+$04] proposes to define an inter-AS routing protocol that runs between specific entities that are responsible for the routing decision on behalf of the IP routers. This could be a way to introduce in a scalable way the notion of QoS into the inter-AS routing protocol. The IPsphere forum [IPs] proposes the addition of a business layer. The SPs advertise the services that they support in the business layer. We note that these services must be composed in order to provide end-to-end services. Before providing a new service between a source and a destination, the list of providers that can be crossed to support this service has to be determined. The complexity of this problem is similar to the complexity of the computation techniques that we have evaluated in this thesis except that the composition may be done in a centralized entity avoiding numerous exchanges between network elements. This complexity depends on the desire to find a set of providers that contribute to the support of an end-to-end service satisfying the requirements or that provide the best end-to-end service satisfying the requirements. This difference in complexity has also been observed between ERO expansion and cooperative PCEs path computation techniques. Finding the best end-to-end service requires to explore the set of all possible compositions while the process of finding a suitable set of providers for a service can be stopped once such a list of providers is found.

Numerous modifications can be brought to the ERO expansion and cooperative PCEs techniques that have been evaluated in this thesis. First, we can consider the use of information gathered in the computation of previously established LSPs in the establishment of new LSPs. This is simpler when the path computation is performed by the nodes along the path. When PCEs participate in the computation of the LSPs, they are not aware of the failure of the establishment of these LSPs if they are not on the path of the LSP. The establishment of LSPs may fail due to changes in the network state and outdated information at the PCE for example. Thus, if the computation is performed by PCEs a mechanism is required for PCEs to learn the LSPs that are really established and eventually the QoS properties along these LSPs. The PCEs and/or the nodes along the LSPs maintain state for the established LSPs. In addition, cooperative PCEs may keep state with the PCRep messages received from previous LSP computations. We note that the computation of LSPs based on these states may lead to the rapid congestion of the resources along established LSPs. With cooperative PCEs, the returned path will not be the shortest path anymore since certain AS-path will not be re-explored. Moreover, the state maintained with the QoS properties of the LSPs and the received PCRep messages for cooperative PCEs, may become outdated. Timers are required to remove these states in order to avoid high failure probability in the establishment of LSPs computed based on the maintained state.

Second, a threshold can be set in order to avoid trying too many downstream paths at a given node in ERO expansion [FSI$^+$05]. This aims at reducing the amount of crankback before deciding that an LSP cannot be established. From our simulations, we deduce that this will probably also reduce the total amount of crankback that occurs in the establishment of a few LSPs. However, the drawback of such threshold is that certain LSPs may not be established even if a suitable path for the LSP exists.

Third, when we impose a limitation on the number of downstream PCEs that can be contacted by a PCE participating in a cooperative computation, certain LSPs cannot be established even if a path respecting the constraints could be found with a complete exploration. We could consider to modify the exploration of paths with cooperative PCEs in order to keep the benefits of a limited exploration for the LSPs that can be established that way and perform a wider exploration when the first exploration has failed. The following issues have to be solved in developing such a solution: (1) a request for a wider exploration cannot be confounded with the first request of exploration for the LSP that may eventually be looping, (2) it should be able to determine when all the possible AS-paths have been explored and the computation process has to end.

In this thesis, we have studied the computation and establishment of LSPs in a stable environment. We did not consider simultaneous requests for LSP establishment. We note that this problem does not seem to have been addressed in the literature on intradomain MPLS TE even if on-line path computations of intra-AS LSPs with pre-emption of lower priority LSPs has been addressed [BML03a].

We have studied the interactions between a load-balancing intra-AS TE algorithm and the provision of delay and bandwidth constrained LSPs, in this thesis. However, many different intradomain TE algorithms have been proposed in the literature to optimize various objectives because different SPs may want to optimize different objectives depending on their business, their network and network management policy. Moreover, it would be interesting to provide services with different requirements than a bandwidth guarantee and a maximum end-to-end delay across AS boundaries. Thus, we only considered a small portion of the possible interactions between intra-AS and inter-AS TE. However, we illustrated in our study that certain objectives are difficult to fulfill simultaneously.

In this thesis, we addressed the problem of providing services across different ASs assuming a single control plane, IP/MPLS. Requirements are now also expressed to provide services across multi-layer and multi-region networks [SPL+06]. An example of a multi-region network is a network composed of IP and optical elements where MPLS controls the IP network elements while GMPLS controls the optical elements. Among the issues that arise when multiple data and control planes are involved, there is the provisioning of services at each layer/region to other layers/regions, the advertisement of these services and their combination into an end-to-end service. Research is lead on providing virtual network topologies and adapting these topologies to the traffic demand and clients requirements [SOIU06]. A virtual topology provides a view of the provider's optical network to the customer. This view is a function of the service to be supported. Underlying resources may be dedicated to each virtual topology. The interactions during the adaptation of different network topologies is a problem that is still to be addressed.

Appendix A

Implementation Description

In section 4.2, we presented different techniques for the computation of constrained inter-AS paths. Now, we give an overview of our implementation of these path computation techniques. We describe the choices that we made concerning points that were not described in the IETF documents at the time of the implementation or are still not described today.

First, we present the objectives for developing the simulator and a set of functionalities provided by our simulator. Then, we describe our implementation of the ERO expansion and Cooperative PCEs path computation techniques.

A.1 Objectives

The ERO expansion and cooperative PCEs path computation techniques have been developed with the desire to quantify the quality of the computed paths and give an idea of the complexity of the different techniques. The objective of the simulator was not to be able to evaluate the robustness of the techniques to network condition changes during the computation and the establishment of LSPs.

A.2 Architecture

We implemented a centralized tool to simulate distributed path computation techniques. Our tool is able to load topologies in different formats. Based on the topology and configuration parameters, it generates a list of LSPs to establish. Then, these LSPs can be established with the centralized path computation technique relying on a global PCE. For the other techniques, the BGP routes need to be computed. For this purpose, our tool generates input files in the format required by C-BGP [QU05]. Different BGP configuration types are possible. Next, our tool launches C-BGP to obtain the BGP routing tables of the nodes. C-BGP also provides us with the classical IP forwarding paths. Our tool parses the BGP routing tables and uses this information in the computation of the paths with the ERO expansion and cooperative PCEs techniques.

There are three common issues to be solved by distributed path computation techniques when the computing nodes have a limited knowledge. First, we have to ensure that the computation will end, that the signalling messages will not endlessly be exchanged between the computing nodes. Secondly, we have to ensure that if a path respecting the constraints for the LSP and the AS policies exists, the path computation technique returns such a path. The third problem is to determine the content of the TED used by the PCE for path computations. The quality of the computed paths and the success in finding a path respecting given constraints depends on the content of the TED. We discuss this issue in chapter 5.

A.3 ERO Expansion

The following problems have to be solved by an implementation of the ERO expansion path computation technique applied to constrained inter-AS path computation: (1) Avoiding loops, (2) Finding a path if it exists, (3) Determining the content of the TED, (4) Selecting a NH when multiple NHs are possible.

The **fi rst problem** to consider when computing paths in a distributed fashion is to avoid the presence of loops in the resulting path. The details of the nodes already crossed by the LSP may not be known by a node that continues the computation of the path for the LSP due to RRO aggregation (see Section 3.6.3) or any other method that enforces the confidentiality requirement expressed in [RV05]. Thus, if this node belongs to a domain that is already crossed by the LSP, loops could occur. The node could compute a segment containing a node already crossed by the LSP. Consequently the establishment of the LSP would fail. There are two solutions to avoid this problem. Either the node is able to contact the node that computed an upstream path segment for the same domain. Then, it asks for the list of nodes along this path segment and computes a segments that avoids these nodes. Or, the second solution is to simply avoid to cross the same domain twice.

A loop could also arise if an egress ASBR does not select a NH that is in another domain. We note that, in such a situation, this node would not be an egress ASBR but a transit node because the LSP does not leave the AS at this node. If such node should decide not to select a NH in another domain, we have to make sure that this node does not select the ingress ASBR that is crossed by the LSP inside the current AS as NH. Moreover, the path segment from this node to the selected NH should be node disjoint from the path segment arriving at the node and contained inside the AS. Looking more closely, we notice that such a situation occurs when the ingress ASBR inside the current domain and the node it selects as egress ASBR do not take a consistent decision. For the ingress ASBR the best way to leave the AS to reach the tail-end of the LSP is through the node it selects as egress ASBR. If there is a better egress ASBR to leave the AS, a path segment should be computed directly to this egress ASBR. Thus, we consider that when an egress ASBR requires the computation of a path segment, this path segment has to leave the current AS.

The **second problem** is to make sure that a path respecting the constraints is found by the algorithm, if such path is possible based on the content of the TED at each PCE. For example, suppose that an LSP from S to D with a $100ms$ maximum delay needs to be established in figure A.1. Assume that when $R41$ received the Path message for the LSP, the RRO contains nodes $S - R21 - R22$. The upstream path for the LSP is $S - R21 - R22$. It learns that the delay along this path is $90ms$. Thus, $R41$ determines that it cannot contribute to compute a path that respects the delay constraint for the LSP. However, if $R41$ is reached via a different upstream path, $S - R31 - R32$, it may be able to contribute successfully to the path computation and establishment. Thus, we have to ensure that $R41$ reevaluates its contribution to the path computation upon reception of a RSVP Path message. In addition, a node may remove certain NHs from consideration if they cause loops in the path resulting from the concatenation of the upstream path and of the path segment to the NH. If no suitable path through the remaining NHs is found for the LSP based on the current upstream path, we have to make sure that if this node is reached through a different upstream path, path segments through all its available NHs will be reevaluated.

The **problem** of determining the content of the TED is not specific to the ERO expansion technique. This problem is discussed in chapter 5.

The **last issue** for inter-AS path computation with the ERO expansion technique is to select an appropriate NH from a set of NHs that do not lead to loops and have not already been tried. The objective of a good selection mechanism is to limit the number of crankback performed during path computation and establishment. This selection is performed based on heuristics because it is not currently possible to know a priori the QoS that can be provided along an interdomain route.

A.3.1 Pseudo Code

A simplified algorithm for ERO expansion is shown in Alg. 1. The first step, $ERO\_expansion\_init()$, consists in the initialization of the variables. The current node, $current\_node$, is set to the source of the LSP. The delay from the source to the current node, $src2current\_delay$, is set to 0. The previous node, $previous\_node$ is set to undef to indicate that there is no upstream ASBR that requested or computed the previous path segment, starting at the upstream ASBR and ending at the current node. The ASN of the source is added to the list containing the set of ASs crossed by the LSP, $crossed\_ASs$. The purpose of this list is to avoid that an LSP crosses an AS multiple times in order to avoid loops. The variable $current\_is\_egress\_ASBR$ indicating if the current node is an egress ASBR is set to false. This variable is useful when the BGP NHs are egress ASBRs (BGP Next-hop self option is used) and not ingress ASBRs inside the downstream domains. The value of this variable indicates if the next ASBR needs to be inside the downstream AS or if it can be inside the current domain. If the current node is an egress ASBR, the computed path contains a segment inside the current AS. We need to leave the AS because we already selected the best egress ASBR. Finally, the source of the LSP is added to the list of nodes representing the computed path, $path$.

After the initialization step we enter into a while loop that ends when the destination of the LSP is reached, that is, when the complete path for the LSP is computed and established. We also leave this loop if the current node is the source of the LSP and all its NHs have been tried without success, i.e. no path respecting the constraints of the LSP has been found.

The objective of each pass through the while loop sequence of instructions is to select the best NH among the suitable NHs for the current node, with regard to the upstream path, the constraints for the LSP and the NHs that have already been tried. If such a NH exist, the path segment to this NH is added to the computed path. If a suitable NH cannot be found crankback occurs.

First, we set the $best\_next\_hop$ and $best\_cost$ variables to the "undef" value, at lines 4 and 5 of Alg. 1. A best NH for the current node is not found yet. Then, function $select\_PCE(current\_node)$ provides the IP address and the index of the PCE responsible for the computation on behalf of the current node, at line 7. This PCE may be the current node itself, a Route-Reflector, a dedicated PCE or any other node.

The block of instructions on lines 8-15 computes path segments to the NHs of the current node and assigns a cost to each segment, if this has not already been done before. If the current node belongs to the same AS as the LSP's tail-end (lines 10-11), a segment from the current node to the tail-end of the LSP is computed based on the topology information distributed by the IGP of the destination AS. Otherwise (lines 12-13), the PCE computes a path segment for each NH that it knows for the LSP's tail-end. These segments start at the current node and end at the NHs. In $compute\_path\_to\_available\_NHs()$ we distinguish two cases: (1) A path to the NH can be computed based on the information distributed by the IGP. This occurs when the NH belongs to the current AS, next-hop self is used, or when it belongs to the downstream AS but the inter-AS TE link information is distributed by the IGP, next-hop self is not used. (2) A path to the NH cannot be computed based on the information distributed by the IGP. This may occur when next-hop self is used. The NH belongs to the neighboring AS. In addition, the inter-AS link TE information is not distributed by the IGP. In this case, the current node is an egress ASBR and it is directly connected to the NH.

In our implementation the path segments to the NHs or to the destination are computed with the Dijkstra algorithm. In case (1), we prune the links that do not have enough reservable bandwidth for the LSP inside the topology of the current AS. With the Dijkstra algorithm, we compute the shortest paths toward the NHs. Here, shortest paths mean the paths with the smallest delays. We keep these path segments in a data structure if their delay is below the end-to-end delay constraint for the LSP. In case (2), the path segment to the NH is the link between the current node and the NH, if there is enough reservable bandwidth on that link. Again, we store this path segment in the data structure if the delay of the link is below the end-to-end delay constraint for the LSP. This data structure is called

$potential\_paths$ .Together with the paths segments, we store their delay/cost in the data structure. In addition, we set the status of the NHs with path segments respecting the bandwidth reservation constraint and with delay below the LSP's end-to-end delay constraint as having to be explored ($TO\_BE\_EXPLORED$). The NHs that cannot be reached with a path segment respecting these constraints are marked as unfeasible ($UNFEASIBLE$). There is also a third status for a NH, the explored status ($EXPLORED$). This indicates that the NH has been tried but did not lead to a feasible solution for the current upstream path; crankback occurred.

Arriving at line 16 of Alg. 1, a delay/cost is assigned to each NH available from the current node or the NH is marked as unfeasible. We note that certain NHs with an assigned cost may have the $EXPLORED$ status. If there in no NH with status $TO\_BE\_EXPLORED$, that is, if all NHs are marked as either unfeasible or explored, the $crankback()$ function is called.

Crankback occurs when there is no NH such that it is possible to complete the upstream path with a path segment that respects the constraints for the LSP, from the current node to the NH. Thus, the $crankback()$ function sets the status of the current node to $EXPLORED$ in the $potential\_paths$ structure attached to the previous node. The $potential\_paths$ structure attached to the current node is removed. This enables to reevaluate the NHs of the current node later, if the current node is reached again from a different upstream path. The $current\_node$ variable is set to the value of $previous\_node$. $previous\_node$ is set to the previous node of $previous\_node$. This information is stored in the $potential\_paths$ data structure. The nodes at the end of the computed path, starting at the current node, are removed from $path$. If certain of the removed nodes do not belong to the same AS as the current node, remove these ASs from the list of crossed ASs, $crossed\_ASs$. Finally, reduce the value of the delay along the upstream path, $src2current\_delay$, from the value of the delay along the removed path segment. If the value of $current\_node$ is undefined (undef) the while loop ends at this stage. All the NHs available at the source have been tried but no path respecting the constraints for the LSP has been found (lines 19-21).

If there is at least one NH with status $TO\_BE\_EXPLORED$ the block starting at line 23 of Alg. 1 is executed. If the best NH has not been selected at line 11, $best\_next\_hop$ is not defined. The purpose of line 25 is to select the best NH according to some heuristic among the NHs that remain to be explored for the current node Among the NHs with status $TO\_BE\_EXPLORED$, the $get\_best\_NH()$ function only considers the NHs that may not lead to loops if used in the path of the LSP. Therefore, if the current node is an egress ASBR, the NH may not be an egress ASBR. This is useful if next-hop self is used. In addition, if the NH is an ingress ASBR, i.e. we enter a new AS, then it cannot belong to an AS that is already crossed by the upstream path. This is determined based on the $crosssed\_ASs$ list.

If the $get\_best\_NH()$ function did not find a suitable NH for the LSP, $best\_next\_hop$ is not defined. Thus, the $crankback()$ function is called at line 29. Again, we note that the while loop may end, if the new value for $current\_node$ is not defined. This occurs if all the NHs available at the source have been tried but no path respecting the constraints for the LSP has been found.

If $get\_best\_NH()$ returns a best NH, $ERO\_expansion\_next\_step()$ is called. The path segment from the current node to the best NH, stored in the $potential\_paths$ data structure of current node, is added to the computed path, $path$. The AS of the current NH is added inside $crossed\_ASs$ if it not already present. The variable $current\_is\_egress\_ASBR$ is set to 1 if the NH is an egress ASBR (useful with next-hop self). It is set to 0 if the NH is an ingress ASBR. $previous\_node$ is set to the current node and the current node is now the best NH. Finally, we add the delay of the path segment to $src2current\_delay$, the delay of the upstream path

After the while loop, line 39, a path has been found for the LSP or there is no path respecting the constraints of the LSP. In the first case, if the LSP requires bandwidth reservation, the amount of reservable bandwidth is reduced by the amount of bandwidth reserved for the LSP on each link used by the LSP. This corresponds to the successful establishment of the LSP. Then, the path and the QoS provided to the LSP are stored in a file. Otherwise, it is mentioned in the file that no path was found for the LSP.

Alg. 1 Simplified algorithm for ERO expansion

```
 1: $ERO\_expansion\_init()$
 2: while $current\_node$ != $dest$ do
 3:    /* The destination is not reached */
 4:    undef $best\_next\_hop$
 5:    undef $best\_cost$
 6:    /* Get the index of the PCE that serves $current\_node$ */
 7:    $(PCE\_IP, PCE\_index) = select\_PCE(current\_node)$
 8:    if ! $has\_been\_met(current\_node)$ then
 9:       /* No info is stored concerning path segments starting at this node */
10:       if $current\_node\_ASN == dest\_AS$ then
11:          $(best\_next\_hop, best\_cost) = compute\_intra\_path\_to\_LSPdest()$
12:       else
13:          $compute\_path\_to\_available\_NHs()$
14:       end if
15:    end if
16:    if $is\_explored(current\_node)$ then
17:       /* $current\_node$ has been explored
          No suitable NH has been found */
18:       $crankback(current\_node)$
19:       if ! defined $current\_node$ then
20:          last
21:       end if
22:    else
23:       if ! defined $best\_next\_hop$ then
24:          /* Certain NHs may have been explored and did not lead to feasible
             solutions
             Compute $best\_next\_hop$ among the remaining NHs */
25:          $(best\_next\_hop, best\_cost) = get\_best\_NH(current\_node)$
26:       end if
27:       if ! defined $best\_next\_hop$ then
28:          /* No suitable NH has been found */
29:          $crankback(current\_node)$
30:          if ! defined $current\_node$ then
31:             last
32:          end if
33:       else
34:          $ERO\_expansion\_next\_step()$
35:       end if
36:    end if
37: end while
38: $reserve\_bandwidth(path)$
```

A.3.2 Example of Execution

The difficulty with ERO expansion is to make sure that all possible paths are tried before asserting that there is no path respecting the constraints of the LSP. In figure A.1, we illustrate this difficulty through the execution of the ERO expansion algorithm, Alg. 1. In this example, an LSP has to be established from head-end S in $AS1$ to tail-end D in $AS6$. This LSP is subject to a $100ms$ delay constraint. We assume that $get\_best\_NH()$ selects the NH reachable with the smallest delay path segments from the current node. Moreover, we suppose that NH self is not used. In order to illustrate the execution of the algorithm, we set a breakpoint between lines 26 and 27 of Alg. 1. We show in figure A.1 the values of the most significant variables at this breakpoint.

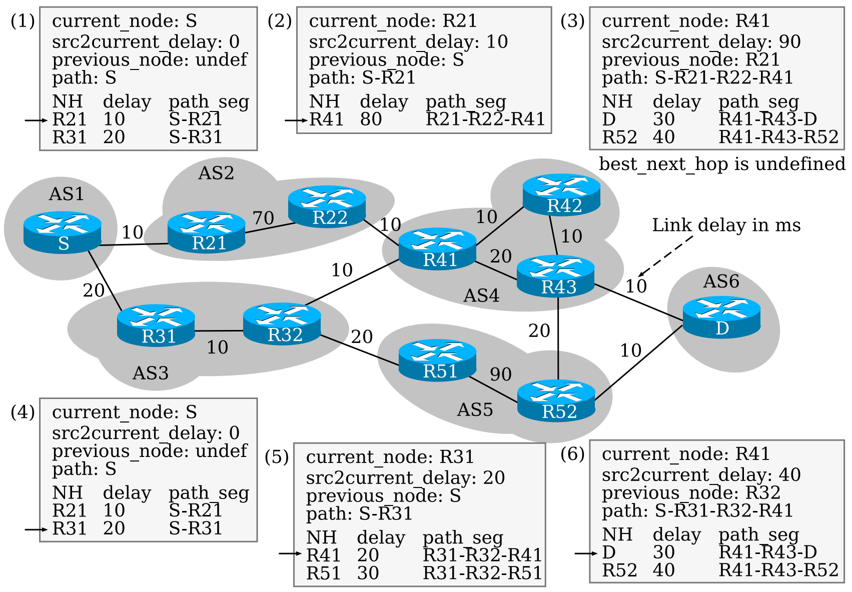

(1)
```
current_node: S
src2current_delay: 0
previous_node: undef
path: S
  NH   delay  path_seg
→ R21  10     S-R21
  R31  20     S-R31
```
(2)
```
current_node: R21
src2current_delay: 10
previous_node: S
path: S-R21
  NH   delay  path_seg
→ R41  80     R21-R22-R41
```
(3)
```
current_node: R41
src2current_delay: 90
previous_node: R21
path: S-R21-R22-R41
  NH   delay  path_seg
  D    30     R41-R43-D
  R52  40     R41-R43-R52
best_next_hop is undefined
```
(4)
```
current_node: S
src2current_delay: 0
previous_node: undef
path: S
  NH   delay  path_seg
  R21  10     S-R21
→ R31  20     S-R31
```
(5)
```
current_node: R31
src2current_delay: 20
previous_node: S
path: S-R31
  NH   delay  path_seg
→ R41  20     R31-R32-R41
  R51  30     R31-R32-R51
```
(6)
```
current_node: R41
src2current_delay: 40
previous_node: R32
path: S-R31-R32-R41
  NH   delay  path_seg
→ D    30     R41-R43-D
  R52  40     R41-R43-R52
```

Figure A.1: ERO expansion execution example

At the beginning of the algorithm, the $current\_node$, $src2current\_delay$, $previous\_node$ and $path$ variables are initialized by the $ERO\_expansion\_init()$ function. Then, we enter the while loop. The PCE to be contacted by the current node, i.e. the head-end of the LSP, is determined. The computed path depends on the information that is available in the TED of the PCE, on the BGP routes present in the TED. Because the current node has not been met before, the path segments to the NHs available in the TED of the PCE are computed (line 13). A cost is assigned to each segment. Here, this cost is the delay of the path segment. The available NHs with path segment below $100ms$ are stored with their delay/cost and the path segment in the $potential\_paths$ structure of the current node. Because no NH of the current node has been explored, we go to line 22. The best NH is not determined yet. Thus, $get\_best\_NH()$ is called and returns $R21$ as the best NH and a value of 10 for the cost of this NH, $best\_cost$. Then, $ERO\_expansion\_next\_step()$ is called to prepare the next iteration of the while loop. The current node is now $R21$. The delay from the source to the current node, $src2current\_delay$, is 10. The previous node is S and the computed upstream path is $S - R21$.

Now, we select the PCE for $R21$ (line 7). Then, we compute the path segments to the NHs for the tail-end of the LSP (line 13). The best NH for $R21$ is determined at line 25. This NH is $R41$. The delay/cost of the segment $R21 - R22 - R41$, from $R21$ to $R41$, is 80. $ERO\_expansion\_next\_step$ sets $current\_node$ to $R41$, $src2current\_delay$ to 90, i.e. the old value of $src2current\_delay$ plus 80. $previous\_node$ is set to $R21$ and $path$ to $S - R21 - R22 - R41$.

Then, we select the PCE for $R41$ (line 7). We compute the path segments to the NHs, D and $R52$, for the tail-end of the LSP (line 13). $get\_best\_NH()$ does not return a NH because the delay of the path segments from the current node to each NH added to the delay of the upstream path, $src2current\_delay$ is above the end-to-end delay constraint of the LSP. Consequently, crankback occurs at line 29. $R41$ is marked as explored in the $potential\_paths$ data structure for node $R21$. This indicates that another NH than $R41$ should be selected for $R21$. The $potential\_path$ data structure for $R41$ is removed. This enables to later try a path through node $R41$ with an upstream path that is different from the current upstream path. The current node becomes node $R21$. $previous\_node$ is set to S, the previous node of node $R21$. The path segment from $R21$ to $R41$ is removed from $path$. Thus, $path$ becomes $S - R21$. $AS4$ is removed from the list of crossed ASs, $crossed\_ASs$. The value for the delay of the upstream path, $src2current\_delay$, is decreased by the delay along the path segments $R21 - R22 - R41$ that is removed from $path$. Its value is again set to $10ms$. Then, we go through the instructions of the while loop once more.

We obtain the PCE serving $R21$ at line 7. We do not need to recompute the path segments for the NHs because this have been done previously. Thus, instructions on lines 9-13 are skipped. $R21$ has a single NH for the tail-end of the LSP. This NH has been explored previously. It is marked with status $EXPLORED$. Thus, condition on line 16 is true. Crankback occurs (line 18). The status of the current node, $R21$, is set to $EXPLORED$ in the $potential\_paths$ data structure of its previous node S. The $potential\_path$ data structure for $R21$ is removed. The current node is now the head-end of the LSP, S and thus, the previous node is undefined. $path$ only contains node S. $AS2$ is deleted from $crossed\_ASs$ list. Finally, $src2current\_delay$ is set to 0. Because $current\_node$ is defined, we do not leave the while loop. Other NHs are tried for the current node.

A PCE is selected for S. The path segments to the NHs of S are not recomputed ($has\_been\_met(S)$ returns true). There is still NH $R31$ that is not explored. In addition, $best\_next\_hop$ is not defined for S. Thus, $get\_best\_NH()$ is called at line 25. This function returns $R31$ as the best NH. The values of the most significant variables at this stage of the execution are provided in frame (4) of figure A.1 Then, $ERO\_expansion\_next\_step()$ is called to prepare the next execution of the while loop instruction block. $current\_node$ is set to the value of

$best\_next\_hop$. $previous\_node$ is set to S. $src2current\_delay$ is set to $20ms$. $AS3$ is added to $crossed\_ASs$ and, finally, the upstream path becomes $S - R31$.

A PCE is selected for node $R31$. The path segments to the NHs available at the PCE for the LSP's tail-end are computed. Segment $R31 - R32 - R41$ with delay of $20ms$ is obtained for NH $R41$ and segment $R31 - R32 - R51$ with delay of $30ms$ is obtained for NH $R51$. Both NHs are still to be explored. $get\_best\_NH()$ selects $R41$ as best NH. The cost/delay to this NH is lower than the cost/delay to NH $R51$. Then, $ERO\_expansion\_next\_step()$ sets the current node to $R41$. We note that the previous node for $R41$ is now $R31$ instead of $R21$. If crankback should occur, another NH for $R31$ should be tried. $AS4$ is added to the $crossed\_ASs$ list. Path segment $R31 - R32 - R41$ is added at the end of $path$. The delay of this path segment is added to $src2current\_delay$.

A PCE is selected for node $R41$. $compute\_path\_to\_available\_NHs()$ is executed because the $potential\_paths$ data structure for $R41$ has been deleted earlier. Consequently all the NHs available for $R41$ can be explored again. This ensures that if a path respecting the constraints exists for the LSP, it will be found. It enables to consider downstream NHs that have not been considered before to avoid loops that arose due to a previous upstream path. NH D is selected by $get\_best\_NH()$, at line 25. This NH can be chosen now because the delay of the resulting upstream path ($70ms$) is below the end-to-end delay constraint for the LSP ($100ms$), contrary to the previous situation. $ERO\_expansion\_next\_step()$ set the current node to D, the previous node to $R41$, the upstream path to $S - R31 - R32 - R41 - R43 - D, \ldots$ Then, the execution of the while loop ends because the LSP's tail-end is reached, i.e. condition at line 2 is false. The bandwidth along the LSP's path is reserved if required for the LSP and the path computed for the LSP is stored in a file together with the QoS properties of this path.

A.3.3 Particularities of the Implementation

In this section we present some assumptions that we made or did not make for our implementation compared to the solution proposed in [VAZ06].

First, we do not assume that the head-end LSR is pre-configured with a list of loose hops or strict abstract nodes to be crossed by LSPs. Thus, the sets of NHs used by the PCEs for path segments computation only depends on the LSP's tail-end. It does not depend on intermediate nodes that need to be crossed. Moreover, the path segments are not constrained to cross specific nodes (except the head-end and the tail-end of the LSP).

[VAZ06] proposes to perform the computation of path segments at the nodes that need to complete the ERO. This occurs when a node receives a Path message where the first node inside the ERO is a loose node or a strict abstract node that is directly connected to the node. Such node is usually at the border of a domain [VAZ06], where the definition of the domain depends on the scope of the computation capability of the nodes. While [VAZ06] assumes the computation of the path segments at nodes that belong to the path of the LSP, we allow these nodes to contact another node, a PCE[1], for the computation.

[RV05] defines an inter-AS MPLS TE LSP as an LSP with head-end and tail-end LSRs that do not belong to the same AS or as an LSP with head-end and tail-end LSRs residing in the same AS but that transits through different ASs. This latter case is not supported by our implementation. If two nodes belong to the same AS, an intra-AS path is computed to connect these two nodes. Our implementation does not allow to enter a domain if it has already been crossed upstream by the LSP.

A.4 Cooperative PCEs

In this section, we provide a description of our implementation of the path computation technique that makes use of Cooperative PCEs. We start with the pseudo code of our implementation. Then, we introduce the particularities of our implementation compared to the solution proposed in [VZB06]. Finally, we illustrate our solution with an example.

A.4.1 Pseudo Code

A simplified algorithm for interdomain path computation by cooperative PCEs is provided in Alg. 2. This algorithm relies on two major data structures: a graph, $path\_segment\_graph$, and a set of explored NHs, $explored\_NHs$. The edges in the graph are local path segments, as defined in 4.1. The vertices of the graph are the head-end and the tail-end nodes of local path segments. As attributes for the edges of the graph, we have the list of nodes along the local path segment and the delay along the path segment. The $explored\_NHs$ set contains the ingress ASBRs for which the local path segments to the available NHs have been computed and the PCEs in the downstream ASs have been contacted.

Alg. 2 Simplified algorithm for Cooperative PCEs

```
Let $path\_segment\_graph$ be the graph composed of the local path segments
Let $explored\_NHs$ be the set of explored NHs
1: $explore\_all\_NHs(src)$
2: $path$ = $compute\_end2end\_path(path\_segment\_graph)$
3: $reserve\_bandwidth(path)$
4: $(nb\_msg\_stateful, nb\_msg\_stateless)$
   $= get\_PCE\_message\_nb(src, paths\_segment\_graph)$
```

First, the function $explore\_all\_NHs()$ is called with the LSP's head-end as parameter, in line 1 of Alg. 2. This function builds the graph composed of the local path segments. It is described in Alg. 3. Then, Dijkstra SPF algorithm is

[1]This node may be the node that requires the path computation, a RR, a dedicated node, ...

run on the graph to find the shortest delay path from the LSP's head-end to tail-end nodes. If such a path exists and its delay is below the maximum end-to-end delay constraint for the LSP, it results a sequence of local path segments. The last action of $compute\_end2end\_path()$ is to transform this sequence of segments into a list of nodes based on the attributes of the edges in $path\_segment\_graph$ in order to obtain the path of the LSP. Then, in line 4 of Alg. 2, bandwidth is reserved for the LSP. The path computed for the LSP and its properties are stored in a file. Finally, the function $get\_PCE\_message\_nb()$ returns a lower and an upper bound on the number of messages exchanged between PCCs and PCEs during the path computation.

Function $explore\_all\_NHs$ is called recursively for each ingress ASBR encountered during the exploration of all the available paths to the LSP's tail-end. A simplified algorithm for this function is provided in Alg. 3. The ingress ASBR is the head of local path segments computed by this call of $explore\_all\_NHs$. It is designated by the variable $segment\_head$. First, the segment head is added inside the set of explored NHs, in line 1. If the segment head is the tail-end of the LSP, this call of the function terminates. A branch of the exploration has arrived to the destination. The $path\_segment\_graph$ contains the possible local path segments for a list of domains between the LSPs head and tail-end nodes. If the segment head is different from the LSP's tail-end, the local path segments starting at $segment\_head$ are computed (line 5). For this purpose, a CSPF algorithm is used on the topology composed of the segment head's AS and the interdomain links connected to the AS. The links with insufficient reservable bandwidth are removed from the topology. It results shortest delay path segments starting at the segment head. Then, if the AS of the segment head is different from the AS of the tail-end node, $get\_all\_BGP\_NHs$ returns all the set of available NHs for the LSP's tail-end (line 7). Then, we explore each NH in the returned set with for loop contained in lines 8 to 22. If the NH is equal to the segment head (line 9), there is no need to explore the NH. Thus another NH from the NHs set is considered. Otherwise (line 12), if a local path segment from the segment head to the NH belongs to $path\_segment\_graph$, another NH is considered. If such a local path segment is not present in the graph, it is added by the function $add\_path\_segment$ (line 15). This function returns true if a suitable local path segment exists between $segment\_head$ and $next\_hop$, with respect to the LSP's bandwidth reservation and end-to-end delay constraint. If such a local path segment does not exist, another NH is explored (line 16). Then, if the NH does not belong to the set of explored NHs, the $explore\_all\_NHs$ function is called recursively with the NH, $next\_hop$, as parameter (line 20). Finally, if the segment head and the tail-end node belong to the same AS, the local path segment from $segment\_head$ to $dest$ is added to the graph with the path segments if it exists and respects the bandwidth and delay constraints of the LSP.

The lower bound, on the number of messages exchanged for the computation of a path with cooperative PCEs, returned by the $get\_PCE\_message\_nb()$ function is the number of adjacencies between ASs in the $path\_segment\_graph$ multiplied

Alg. 3 Simplified algorithm for $explore\_all\_NHs(segment\_head)$

```
1: $explored\_NHs\_add(segment\_head)$
2: if $segment\_head == dest$ then
3:     return
4: end if
5: $sssp = compute\_local\_path\_segments(segment\_head)$
6: if ! $segment\_head\_ASN == dest$ then
7:     $NHs = get\_all\_BGP\_NHs(segment\_head)$
8:     for all $i$ in $NHs$ do
9:         if $next\_hop == segment\_head$ then
10:            next
11:        end if
12:        if $path\_segment\_graph.has\_edge(segment\_head, next\_hop)$ then
13:            next
14:        end if
15:        $exist\_segment = add\_path\_segment($
           $path\_segment\_graph, sssp, segment\_head, next\_hop$
           $)$
16:        if ! $exist\_segment$ then
17:            next
18:        end if
19:        if ! $is\_explored\_NH(next\_hop)$ then
20:            $explore\_all\_NHs(next\_hop)$
21:        end if
22:    end for
23: else
24:    $add\_path\_segment(path\_segment\_graph, sssp, segment\_head, dest)$
25: end if
```

by two. If there are path segments from ingress ASBRs in ASx to ingress ASBRs in ASy in $path\_segment\_graph$, at least one PCReq message has been sent from the PCE covering ASx to the PCE covering ASy. Moreover, for each PCReq message sent, a PCRep or a PCErr is received in reply.

The algorithm to compute the upper bound on the number of messages exchanged between cooperative PCEs is provided in Alg. 4. It relies on the AS adjacency graph built from $path\_segment\_graph$ for the computation of the lower bound on the number of messages exchanged. It consist on an exploration of all the paths, in terms of the domains crossed, that can be used from the LSP's head to tail-end nodes. We count the number of messages that are propagated along these paths.

Alg. 4 Simplified algorithm for $get\_stateless\_PCE\_message\_nb()$

```
Let $current\_AS$ be the ASN of the current AS (parameter)
Let $AS\_adj$ be the structure with the adjacencies between ASs in $path\_segment\_graph$ (parameter)
Let $crossed\_ASs$ be the set of ASs crossed by the current branch (parameter)
1:  $nb\_messages = 0$
2:  $local\_crossed\_ASs = crossed\_ASs$
3:  $add\_crossed\_AS(local\_crossed\_ASs, current\_AS)$
4:  for all $downstream\_AS$ in $adjacent\_to(AS\_adj, current\_AS)$ do
5:    if ! $is\_crossed(local\_crossed\_ASs, downstream\_AS)$ then
6:      /* Otherwise the request is looping */
7:      $nb\_messages += 1 + get\_stateless\_PCE\_message\_nb($
        $downstream\_AS,$
        $AS\_adj,$
        $local\_crossed\_ASs)$
8:    else
9:      /* Message is sent to downstream AS but not further */
10:     $nb\_messages += 1$
11:   end if
12: end for
13: return $nb\_messages * 2$
```

A.4.2 Particularities of the Implementation

Our implementation differs from the technique proposed in [VZB06] in a few points. These divergences do no have an impact on the resulting computed paths but on the way these paths are computed and on the information exchanged between cooperating PCEs. These differences are introduced in this section.

First, in BRPC [VZB06], a PCE replies to a PCReq with path segments that start at an ingress node of its domain and end at the tail-end of the LSP. To achieve this, the PCEs build a topology from the received path segments, the local topology and the inter-AS links. They perform a CSPF computation on this topology. In my implementation, the PCEs do not perform this computation. The local segments computed by a PCE are transmitted as is to the upstream PCE until the PCE with domain containing the head-end LSR. This PCE performs a CSPF computation on the topology composed of the path segments received from the downstream PCEs. This implies that the size of the PCRep messages in my implementation is bigger than in BRPC. All the local path segments are sent instead of aggregating them with downstream paths segments to obtain path segments to the destination as in BRPC.

Second, in BRPC, a PCE computes the local path segments upon reception of the replies of the PCReq that it sent. In my implementation, the local path segments are computed on receipt of a PCReq message. If there is no local path segment respecting the constraints for the LSP toward a NH, then no downstream path will be computed for this NH. For example, considering a bandwidth reservation and a maximum end-to-end delay for the LSP, if there is no local path segment from ingress ASBRs to the NH with enough available bandwidth for the LSP and with delay below the delay constraint for the LSP, no downstream path is computed for the NH. For this purpose, we assume that a PCE specifies the head-end nodes of the path segments it requests from a downstream PCE. Instead of computing path segments from all ingress ASBRs connected to a given upstream AS, a PCE computes local path segments only from a subset of ingress ASBRs specified in the PCReq message. A consequence of this difference is that, in our implementation, if there is a single NH in a downstream AS and there is no suitable local path segment to reach the NH, the PCE in this AS is not contacted. In BRPC, the downstream PCE would be contacted anyway. However, we will see later in this section that there are situations where this difference may result in more messages being exchanged with my implementation than with BRPC.

Third, I do not assume in my implementation that the list of domains to be crossed by the LSP is known prior to the path computation. Thus, in my implementation, a PCE receiving a PCReq for an LSP sends PCReq messages to all the PCEs in downstream domains containing at least one reachable NH (learned from the BGP routes). Two issues result from this particularity. First, a PCE has to be able to detect duplicated PCReq messages from messages that are received from a different list of upstream PCEs. For this purpose, we consider that PCReq messages contain the list of identifiers for the PCEs[2] which have been crossed by the request. If a PCE receives a request with its identifier in the list, it sends a Path Computation Error (PCErr) message to the PCE that sent the request. Upon reception of a PCErr message a PCE does not wait to receive path information from the downstream PCE that sent this message. Thus, it can form a reply to its PCC if it received the answers for the other PCReq messages that it generated.

A second issue to consider concerns the state to be maintained by a PCE. A PCE may receive multiple PCReq for the establishment of the same LSP. These PCReq may be received from the same upstream PCE with different lists of PCE identifiers or they may be received from different upstream PCEs. We distinguish two types of PCEs with respect to the state maintained by the PCEs. First, we consider PCEs that do not keep state on the computation performed, upon reception of a PCReq for an LSP, after having replied to the PCReq. We call such a PCE a “stateless” PCE. When a stateless PCE receives a PCReq message, it sends PCReq messages to the downstream PCEs. The PCE waits for all the PCRep of the downstream PCEs before computing and sending its own PCRep. During that time, the stateless PCE keeps some information for the LSP: the local path segments and the segments already received in the PCRep from the downstream PCEs. This is

[2]This list may contain identifi ers for the domains of the PCEs instead of PCE identifi ers. This may allow to determine if a request has already been seen by another PCE in the domain when multiple PCEs are present in a domain.

the minimum amount of information required by the PCE for the path computation. Once the stateless PCE sends its PCRep it removes the information that was maintained for the LSP. Thus, a stateless PCE treats received PCReq messages independently from previously received PCReq messages. However, PCE may maintain the local path segments computed for previous PCReq in its state and the responses from the previous downstream PCEs in its state. Such a PCE does not need to recompute already known local path segments. Moreover, it only sends PCReq messages to the PCEs that did not participate in previous computations for the available NHs. Some PCEs may still need to be contacted due to the unavailability of local path segments from earlier ingress ASBRs to NHs belonging to their domains, that respect the constraints of the LSP, in my implementation, or due to loop prevention concerning earlier requests, in both BRPC and my implementation.

Figure A.2 illustrates the fact that the values returned by the $get\_PCE\_message\_nb()$ function are lower and upper bounds on the number of messages exchanged between PCCs and PCEs for my implementation of the cooperative PCEs path computation technique. The edges in the figure are all directed from the head-end S to the tail-end D of the LSP. In part (a) of the figure, we have a simplified view of the path segments graph, $path\_segment\_graph$, at the end of the execution of Alg. 2. Part (b) of the figure shows the AS adjacency graph corresponding to the path segment graph of part (a). In part (c) we find the PCReq messages that are exchanged to obtain the path segment graph of (a). We note that there is no suitable local path segment between ingress ASBR $R41$ and NH $R52$. Thus, PCReq message (3) generated upon receipt of PCReq (2) only asks for segments starting at $R51$. When $PCE4$ receives PCReq (5), NH $R52$ becomes reachable from a suitable local path segment starting at $R43$. Therefore, $PCE4$ sends another PCReq, PCReq (6) to $PCE5$. This message requests segments starting at $R52$. We observe that 6 PCReq messages are generated while there are 5 edges in the AS adjacency graph. Moreover, there are two different AS paths from $AS1$ to $AS5$: $AS1 - AS2 - AS4 - AS5$ and $AS1 - AS3 - AS4 - AS5$. It results an upper bound of 6 PCReq messages; three messages are exchanged along both AS paths.

The lower bound computed by the $get\_PCE\_message\_nb()$ function is also a lower bound on the number of PCEP messages with BRPC. With BRPC, five messages would be exchanged with stateful PCEs, in the example of figure A.2. The lower bound is achieved in this example. In BRPC, $PCE5$ would answer with path segments starting a $R51$ and $R52$ to PCReq (3) received from $PCE4$. Thus, if this PCE maintains a state with these segments it will not need to recontact $PCE5$ when it receives PCReq (5) from $PCE3$. However, with stateless PCEs $PCE4$ needs to send a second PCReq to $PCE5$ when it receives PCReq (5). Here the upper bound on the number of messages is achieved.

The upper bound that is computed for our implementation is not an upper bound on the number of messages that may be exchanged with BRPC. There are situations where BRPC may require more messages than the value that is returned by the $get\_PCE\_message\_nb()$ function. For example, in figure 4.4 in chapter 4, 10 messages are required to compute the LSP from S to D with a 100 ms delay constraint. However, we see in figure A.3 that only 8 messages are required to compute a path for this LSP on the same topology, with our implementation. Moreover, $get\_PCE\_message\_nb()$ returns an upper bound value of 8 because there is no edge between $AS2$ and $AS4$ in the AS adjacency graph obtained from my implementation. We have an example where the upper bound that is computed for the technique that I implemented is not an upper bound on the number of messages exchanged with BRPC.

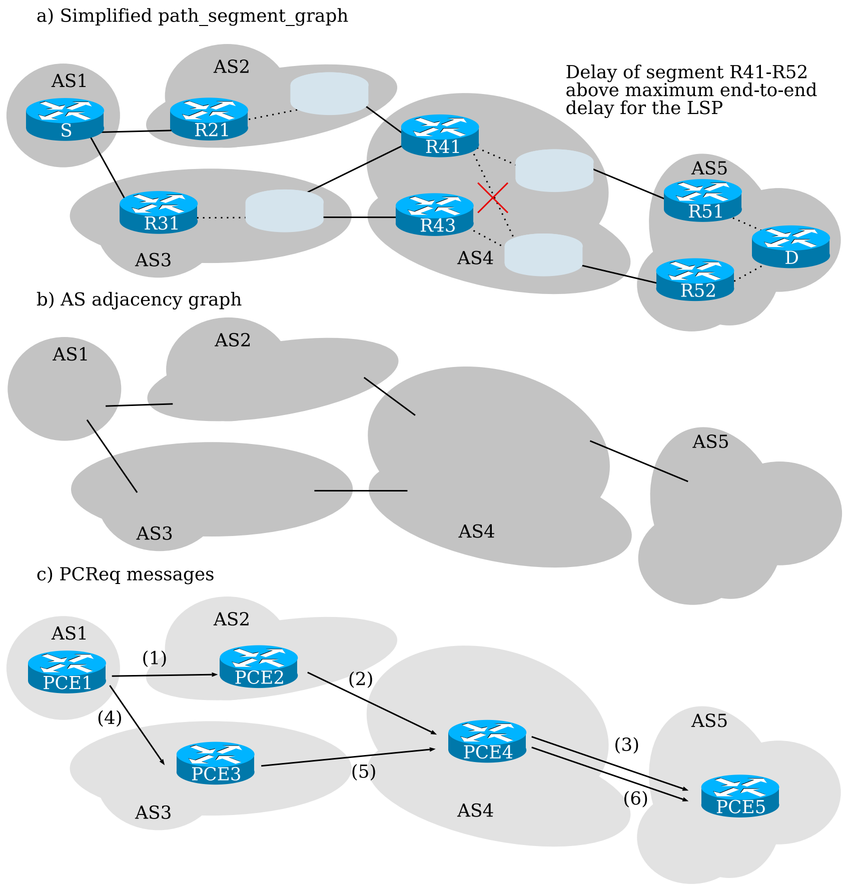
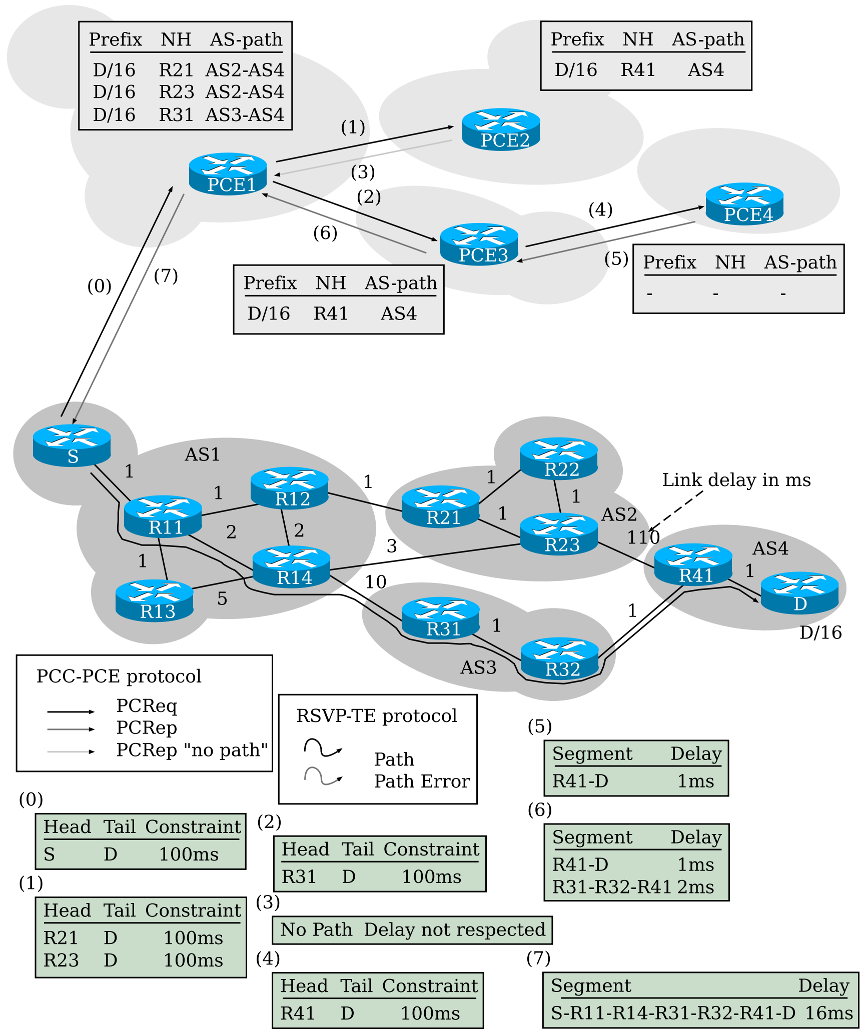


Figure A.2: Lower bound on number of messages exchanged between cooperative PCEs

A.4.3 Example

In figure A.3, we illustrate the exchange of PCReq and PCRep messages for the computation of a path assuming that our algorithm is distributed on multiple PCEs.

The same LSP as in figure 4.4 is to be established.



Prefix	NH	AS-path
D/16	R21	AS2-AS4
D/16	R23	AS2-AS4
D/16	R31	AS3-AS4

Prefix	NH	AS-path
D/16	R41	AS4

Prefix	NH	AS-path
D/16	R41	AS4

Prefix	NH	AS-path
-	-	-

(0)

Head	Tail	Constraint
S	D	100ms

(1)

Head	Tail	Constraint
R21	D	100ms
R23	D	100ms

(2)

Head	Tail	Constraint
R31	D	100ms

(3)

No Path Delay not respected

(4)

Head	Tail	Constraint
R41	D	100ms

(5)

Segment	Delay
R41-D	1ms

(6)

Segment	Delay
R41-D	1ms
R31-R32-R41	2ms

(7)

Segment	Delay
S-R11-R14-R31-R32-R41-D	16ms

Figure A.3: Cooperative PCEs

The head-end of the LSP, S sends a PCReq message to the PCE of its AS, $PCE1$. $PCE1$ in $AS1$ has three routes for prefix $D/16$. Two of these routes are received from $AS2$, with two different BGP Next-Hops (NHs) and the other route is received from $AS3$. Thus, $PCE1$ computes three paths segments starting at S and ending at each of the different NHs in the downstream ASs. Then, it sends a PCReq message to $PCE2$ and $PCE3$, the PCEs inside $AS2$ and $AS3$. The PCReq messages contain a list of head-end nodes for the requested path segments, the address of the tail-end of the LSP and the delay constraints of the LSP. For example, message (1) indicates that PCE1 requests two paths segments: one from $R21$ to D and one from $R23$ to D. The end-to-end maximum delay constraint is specified in the PCReq messages instead of the end-to-end delay of the path that remains to be computed. Otherwise, certain local path segments could be removed from consideration because their delay is above the delay constraint of the LSP minus the delay of the upstream path. However such path segment could be suitable if the AS is reached from a different upstream path with lower delay.

There is no path segment with delay below the $100ms$ constraints from $R21$ and $R23$ to the BGP NH $R41$. Thus, $PCE2$ sends a PCRep "no path" message to its client $PCE1$. $PCE3$ has one route for prefix $D/16$. The NH for this route is router $R41$ in $AS4$. Therefore, $PCE3$ computes a path segment from the only head-end node $R31$, received in PCReq (2) from $PCE1$, to $R41$.

Then, $PCE3$ requests $PCE4$ for a path segment from $R41$ to D. This PCE computes a path segment from $R41$ to D and sends the segment in PCRep message (5) upstream to $PCE3$. When $PCE3$ receives the PCRep, it has the response to the single PCReq that it sent for the LSP. It has all the requested information. Next, $PCE3$ sends the local path segment and the received path segments inside PCRep (6) to $PCE1$. The domain of $PCE1$ contains the LSP's head-end. Thus, upon reception of replies from all the PCEs it sent PCReq messages to, it computes the end-to-end path based on the local and received segments. It builds a graph from these segments and performs a SPF[3] computation on this graph. Since the resulting path respects the constraints for the LSP, it is sent in PCRep (7) to the head-end of the LSP, S. Finally, S initiates the establishment of the LSP along this path.

In this example, the computation that takes place at the PCEs, for each entrance inside the domain of the PCE, consists in a Shortest Path First (SPF) algorithm with the delay as metric. If the delay of all the resulting local path segments is too high, a PCRep "no path" message is sent to the PCC.

A.5 Conclusion

In this appendix, we introduced the difficulties in implementing the ERO expansion and cooperative PCEs techniques for inter-AS constrained path computations. These concern loop detection mechanisms for both the RSVP-TE and PCEP signalling protocols while being able to find a suitable path for the LSP if such a path exist. It has to make sure that all the possible AS-paths are considered in the computation before deducing that a suitable path cannot be provided. This requires to pay attention to the way the AS-paths are traversed by our implementation. Depending on the information maintained inside PCEs and our implementation, the exploration of several AS-path may be avoided. The price for this limited exploration is in the amount of memory required. Thus, we had to deal with the widely known trade-off between the memory and time consumption in our implementation.

We described the different choices that we made for the implementation. We have shown that these choices do not impact on the quality of the computed paths.

[3]Another algorithm could be used

However, they have an influence on the information that is exchanged between PCEs and on the number of messages exchanged between PCEs.

Appendix B

BGP Next-Hop Self and Diversity

In the simulations of chapter 5, the ASBRs were configured with the BGP next-hop self option. This means that when an ASBR learns a route on an eBGP session, it changes the NH advertised with the route. It replaces the NH by its address. The usage of this option has an impact on diversity of the routes that can be computed by the DPC technique described in chapter 5.

In figure B.1, we give an example of distributed path computation without the use of the next-hop self option. Routers $R13$ and $R14$ both select the route advertised by $R22$ for prefix $130.104/16$. They advertise this route without changing the NH because the next-hop self option is not activated. The source (PE) node receives two routes with NH set to a different interface of $R22$. It selects one of these routes as the best route. Because the NHs learned at the source PE are in the downstream AS, the source PE computes a path till the entrance inside the downstream AS. These two NHs are interface of $R22$. Thus, the source PE establishes the protected path to this node. Upon reception of the Path message, $R22$ completes the path to the destination PE. We note that nodes $R14$ and $R13$ do not participate in the computation of the path. The source PCE cannot compute a path that is disjoint from the protected path $Source - R12 - R14 - R22 - R23 - Dest$. Consequently, the primary LSP cannot be protected even if a disjoint path exists and is learned by $R13$.

Through this example, we noticed that the fact of not using the BGP next-hop self option, when a BGP router has multiple eBGP session[1], may reduce the diversity of the paths available to a distributed path computation technique. However it also reduces the number of nodes that participate in the computation.

[1]This does not occur in the topologies used in chapter 5.

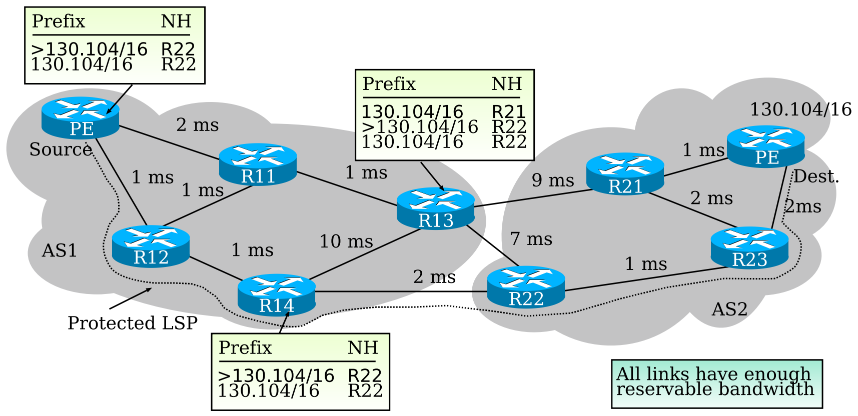


Figure B.1: Diversity without BGP next-hop self option

Bibliography

[ABG$^+$01] D. Awduche, L. Berger, D.-H. Gan, T. Li, V. Srinivasan, and G. Swallow. RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC 3209, December 2001.

[ABKM01] D. G. Andersen, H. Balakrishnan, M. Frans Kaashoek, and R. Morris. The case for resilient overlay networks. In *Proc. of the 8th Annual Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, May 2001.

[ACE$^+$02] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of internet traffic engineering. RFC 3272, May 2002.

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

[ADF$^+$01] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas. LDP specification. RFC 3036, January 2001.

[AKK$^+$00] P. Aukia, M. Kodialam, P. Koppol, T. Lakshman, H. Sarin, and B. Suter. RATES: A server for MPLS Traffic Engineering. *IEEE Network Magazine*, pages 34–41, March/April 2000.

[AKNS06] R. Aggarwal, K. Kompella, T. Nadeau, and G. Swallow. BFD for MPLS LSPs. Internet draft, draft-ietf-bfd-mpls-03.txt, work in progress, June 2006.

[AL06] G. Ash and J-L. Leroux. PCE communication protocol generic requirements. Internet draft, draft-ietf-pce-comm-protocol-gen-reqs-06.txt, work in progress, May 2006.

[AMA$^+$99] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over MPLS. RFC 2702, September 1999.

[AU99] A. Antony and H. Uijterwaal. Routing Information Service - R.I.S. - design note. RIPE NCC document RIPE-200, http://www.ripe.net/ripe/docs/RIS/, October 1999.

[AV06a] A. Ayyangar and J-P. Vasseur. Inter domain GMPLS Traffic Engineering - RSVP-TE extensions. Internet draft, draft-ietf-ccamp-inter-domain-rsvp-te-03.txt, work in progress, March 2006.

[AV06b] A. Ayyangar and J-P. Vasseur. Label Switched Path stitching with Generalized MPLS Traffic Engineering. Internet draft, draft-ietf-ccamp-lsp-stitching-03.txt, work in progress, March 2006.

[Bar02] J. Bartlett. Optimizing multi-homed connections. *Business Communications Review*, 32(1):22–27, January 2002.

[BBC$^+$98] S. Blake, D. Black, M. Carlson, E. Davies, and Z. Wangand W. Weiss. An architecture for differentiated services. RFC 2475, December 1998.

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

[BCC00] T. Bates, R. Chandra, and E. Chen. BGP route reflection - an alternative to full mesh iBGP. Internet RFC 2796, April 2000.

[BCC06] T. Bates, R. Chandra, and E. Chen. BGP route reflection - an alternative to full mesh IBGP. Internet draft, draft-ietf-idr-rfc2796bis-10.txt, work in progress, March 2006.

[BCED$^+$] S. Balon, S. Cerav-Erbas, O. Delcourt, J. Lepropre, G. Monfort, B. Quoitin, F. Skivée, and H. Umit. TOTEM 2.1 - user guide. `http://totem.run.montefiore.ulg.ac.be`.

[BCS94] R. Braden, D. Clark, and S. Shenker. Integrated services in the Internet architecture : an overview. RFC 1633, July 1994.

[BFF05] O. Bonaventure, C. Filsfils, and P. François. Achieving sub50 milliseconds recovery upon BGP peering link failures. In *Proceedings of CoNEXT 2005*, October 24-27th 2005.

[BGT02] T. Bu, L. Gao, and D. Towsley. On characterizing BGP routing table growth. In *IEEE Global Internet 2002*, November 2002.

[Bha99] R. Bhandari. *Survivable Networks: Algorithms for diverse routing*. Kluwer Academic Publishers, 1999.

[BML03a] F. Blanchy, L. Mélon, and G. Leduc. A Preemption-Aware On-line Routing Algorithm for MPLS Networks. *Telecommunication Systems*, 24:187–206, 2-4, Oct.-Dec. 2003.

[BML03b] F. Blanchy, L. Mélon, and G. Leduc. An efficient decentralized on-line traffic engineering algorithm for MPLS networks. In J. Charzinski, R. Lehnert, and P. Tran-Gia, editors, *Proc. of 18th International TELETRAFFIC CONGRESS - Providing Quality of Service in Heterogeneous Environments*, volume 5a, pages 451–460, Berlin, Germany, 31 Aug.-5 Sep. 2003.

[Bon04] O. Bonaventure. LISIS - a simple ISIS analyser. `http://totem.info.ucl.ac.be/lisis_tool/`, 2004.

[BOR$^+$02] A. Basu, C. Luke Ong, A. Rasala, F. Bruce Shepherd, and G. Wilfong. Route oscillations in I-BGP with route reflection. In *SIGCOMM'02*, Pittsburgh, PA, USA, August 2002.

[Bou05] M. Boucadair. QoS-Enhanced Border Gateway Protocol. Internet draft, draft-boucadair-qos-bgp-spec-01.txt, work in progress, July 2005.

[BPSA01] M. Blanchet, F. Parent, and B. St-Arnaud. Optical BGP (OBGP): InterAS lightpath provisioning. Internet draft, draft-parent-obgp-01.txt, work in progress, March 2001.

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

[BVF06] R. Bradford, J-P. Vasseur, and A. Farrel. Preserving topology confidentiality in inter-domain path computation and signaling. Internet draft, draft-bradford-pce-path-key-00.txt, work in progress, June 2006.

[BZB$^+$97] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reSerVation Protocol (RSVP) – version 1 functional specification. RFC 2205, September 1997.

[CB96] E. Chen and T. Bates. An application of the BGP community attribute in multi-home routing. RFC 1998, August 1996.

[CC04] M. Carugi and J. De Clercq. Virtual Private Network Services: Scenarios, Requirements and Architectural Constructs from a Standardization Perspective. *IEEE Communications Magazine*, 42(6), June 2004.

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

[CGJ+04] H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Towards Capturing Representative AS-Level Internet Topologies. *Computer Networks Journal, Elsevier*, 44(6):737–755, April 2004.

[CJ03] G. Cristallo and C. Jacquenet. Providing quality of service indication by the BGP-4 protocol: the QOS_NLRI attribute. Work in progress, draft-jacquenet-qos-nlri-05.txt, June 2003.

[CK06] M. Crovella and B. Krishnamurthy. *Internet Measurements - infrastructure, traffic and applications*. Wiley, 2006.

[CP04] S. De Cnodder and C. Pelsser. Protection for inter-AS MPLS tunnels. Internet draft, draft-decnodder-ccamp-interas-protection-00.txt, work in progress, July 2004.

[dBL03] C. de Launois, O. Bonaventure, and M. Lobelle. The NAROS approach for IPv6 multihoming with traffic engineering. In *4th COST 263 International Workshop on Quality of Future Internet Services (QoFIS 2003)*, volume LNCS 2811, pages 112–121, Stockholm, Sweden, October 1-3rd 2003. Springer-Verlag.

[DCKM04] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of the ACM SIGCOMM '04 Conference*, Portland, Oregon, August 2004.

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

[dUB05a] C. de Launois, S. Uhlig, and O. Bonaventure. Scalable route selection for IPv6 multihomed sites. In *Proceedings of Networking 2005*, Waterloo, Ontario, Canada, May 2-6th 2005.

[dUB05b] C. de Launois, S. Uhlig, and O. Bonaventure. Scalable route selection for IPv6 multihomed sites. In *Proceedings of Networking 2005*, Waterloo, Ontario, Canada, May 2-6th 2005.

[EJLW01] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS Adaptive Traffic Engineering. In *IEEE INFOCOM 2001*, Anchorage, Alaska, April 22-26th 2001.

[FB05] P. François and O. Bonaventure. Avoiding transient loops during IGP convergence in IP networks. In *Proc. of IEEE INFOCOM 2005*, March 2005.

[FBR03] N. Feamster, J. Borkenhagen, and J. Rexford. Guidelines for interdomain traffic engineering. *SIGCOMM Computer Communication Review*, 33(5):19–30, 2003.

[FBR$^+$04] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe. The case for separating routing from routers. In *ACM SIGCOMM workshop on Future Directions in Network Architecture (FDNA 2004)*, August 2004.

[FE05] C. Filsfils and J. Evans. Deploying DiffServ in IP/MPLS backbone networks for tight SLA control. *IEEE Internet Computing*, 9(1):58–65, 2005.

[FFEB05] P. François, C. Filsfils, J. Evans, and O. Bonaventure. Achieving subsecond IGP convergence in large IP networks. *ACM SIGCOMM Computer Communications Review*, 35(3):35–44, 2005.

[FSI$^+$05] A. Farrel, A. Satyanarayana, A. Iwata, N. Fujita, and G. Ash. Crankback signaling extensions for MPLS and GMPLS RSVP-TE. Internet draft, draft-ietf-ccamp-crankback-05.txt, work in progress, May 2005.

[FT00] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *INFOCOM2000*, March 2000.

[FT02] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, 2002.

[FT03] B. Fortz and M. Thorup. Robust optimization of OSPF/IS-IS weights. In W. Ben-Ameur and A. Petrowski, editors, *Proc. INOC 2003*, pages 225–230, October 2003.

[FT04] B. Fortz and M. Thorup. Increasing internet capacity using local search. *Computational Optimization and Applications*, 29(1):13–48, 2004.

[FVA05] A. Farrel, J-P. Vasseur, and A. Ayyangar. A framework for interdomain MPLS traffic engineering. Internet draft, draft-ietf-ccamp-inter-domain-framework-04.txt, work in progress, July 2005.

[FVA06] A. Farrel, J.-P. Vasseur, and J. Ash. A Path Computation Element (PCE) based architecture. RFC 4655, August 2006.

[Gao00] L. Gao. On inferring autonomous system relationships in the internet. In *Proc. IEEE Global Internet Symposium*, November 2000.

[GCL04] K. Gopalan, T. Chiueh, and Y. Lin. Load Balancing Routing with Bandwidth-Delay Guarantees. *IEEE Communications Magazine*, 42(6), June 2004.

[GDZ05] R. Gao, C. Dovrolis, and E. Zegura. Interdomain ingress traffic engineering through optimized AS-path prepending. In *Proceedings of IFIP Networking 2005*, May 2005.

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

[GW02a] T. Griffin and G. Wilfong. Analysis of the MED oscillation problem in BGP. In *ICNP2002*, November 2002.

[GW02b] T. Griffin and G. Wilfong. On the correctnes of iBGP configuration. In *SIGCOMM'02*, pages 17–29, Pittsburgh, PA, USA, August 2002.

[H^+05] M. Howarth et al. Provisioning for interdomain quality of service: the MESCAL approach. *IEEE Communications Magazine*, pages 129–137, June 2005.

[Hal00] B. Halabi. *Internet Routing Architectures (2nd Edition)*. Cisco Press, 2000.

[HFP^+02] Bradley Huffaker, Marina Fomenkov, Daniel J. Plummer, David Moore, and k claffy. Distance metrics in the internet. In *IEEE International Telecommunications Symposium*, 2002.

[HFP^+05] M.P. Howarth, P. Flegkas, G. Pavlou, N. Wang, P. Trimintzios, D. Griffin, J. Griem, M. Boucadair, P. Morand, H. Asgari, and P. Georgatsos. Provisioning for inter-domain quality of service: the mescal approach. *IEEE Communications Magazine*, June 2005.

[Hus01] G. Huston. Analyzing the internet's BGP routing table. *Internet Protocol Journal*, 4(1), March 2001.

[Hus06a] G. Huston. AS1221 BGP table data. `http://bgp.potaroo.net/1221/bgp-active.html`, October 2006.

[Hus06b] G. Huston. Cidr report for 14 aug 06. `http://www.cidr-report.org/`, August 2006.

[HWPTH04] K-H. Ho, N. Wang, G. Pavlou P. Trimintzios, and M. Howarth. On egress router selection for inter-domain traffic with bandwidth guarantees. In *IEEE Workshop in High Performance Switching and Routing (HPSR'2004)*, pages 337 – 342, Arizona, USA, April 18-21st 2004.

[IPs] IPsphere FORUM - the business of IP. `http://www.ipsphereforum.org/home`.

[Jac03] C. Jacquenet. Providing quality of service indication by the BGP-4 protocol : the QoS_NLRI attribute. Internet draft, draft-jacquenet-qos-nlri-05.txt, work in prog ress, June 2003.

[KHC+06] A. Kvalbein, A. Hansen, T. Cicic, S. Gjessing, and O. Lysne. Fast ip network recovery using multiple routing configurations. In *Proc. of IEEE INFOCOM 2006*, April 2006.

[KKY03] D. Katz, K. Kompella, and D. Yeung. Traffic Engineering (TE) extensions to OSPF version 2. RFC 3630, September 2003.

[KR05] K. Kompella and Y. Rekhter. Label Switched Paths (LSP) hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE). RFC 4206, October 2005.

[KYGS05] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. IP fault localization via risk modeling. In *2nd Symposium on Networked Systems Design & Implementation (NSDI'05)*, 2005.

[LABJ00] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. In *SIGCOMM 2000*, August 2000.

[LC05] S. Lo and R. K. C. Chang. Active measurement of the AS Path prepending method. In *IEEE International Conference on Network Protocols (ICNP'2005)*, November 2005. Poster paper.

[LFC05] C.Y. Lee, A. Farrel, and S. De Cnodder. Exclude routes - extensions to RSVP-TE. Work in progress (Publication requested), draft-ietf-ccamp-rsvp-te-exclude-route-05.txt, August 2005.

[LMO+06] J-L. Le Roux, P. Mabey, E. Oki, R. Rabbat, T. Wo Chung, and R. Zhang. Requirements for path computation element PCE discovery. Internet draft, draft-ietf-pce-discovery-reqs-05.txt, work in progress, June 2006.

[LVB05] J-L. Le Roux, J-P. Vasseur, and J. Boyle. Requirements for inter-area MPLS Traffic Engineering. RFC 4105, June 2005.

[McM99] P. McManus. A passive system for server selection within mirrored resource environments using AS path length heuristics. Available from `http://www.gweep.net/~mcmanus/proximate.pdf`, April 1999.

[MD03] A. Sridharanand S. Moon and C. Diot. On the correlation between route dynamics and routing loops. In *IMC '03: Proceedings of the*

3rd ACM SIGCOMM conference on Internet measurement, pages 285–294, 2003.

[ML05] I. Minei and J. Lucek. *MPLS-Enabled Applications*. Wiley, 2005.

[MSWA02] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. Inferring link weights using end-to-end measurements. In *2nd Internet Measurement Workshop (IMW2002)*, Marseille, France, November 6-8th 2002.

[NEN02] I. Nakagawa, H. Esaki, and K. Nagami. A design of a next generation IX using MPLS technology. In *2002 Symposium on Applications and the Internet (SAINT'02, IEEE)*, Nara, Japan, Jan 28th-Feb 1st 2002.

[OHMC01] I. T. Okumus, J. Hwang, H. A. Mantar, and S. J. Chapin. Interdomain LSP setup using bandwidth management points. In *IEEE Globecom*, San Antonio, Texas, 25-29 November 2001.

[PB02] C. Pelsser and O. Bonaventure. RSVP-TE extensions for interdomain LSPs, October 2002. Work in progress, draft-pelsser-rsvp-te-interdomain-lsp-00.txt.

[PB03] C. Pelsser and O. Bonaventure. Extending RSVP-TE to support inter-AS LSPs. In *2003 Workshop on High Performance Switching and Routing (HPSR 2003)*, June 24-27th 2003.

[PQB03] C. Pelsser, B. Quoitin, and O. Bonaventure. Distribution of QoS information with BGP. D4.3 Deliverable: Synthesis and Recommendations for Interdomain Traffic Engineering, November 2003. IST-1999-20675-Atrium.

[PSA05] P. Pan, G. Swallow, and A. Atlas. Fast reroute extensions to RSVP-TE for LSP tunnels. RFC 4090, May 2005.

[PUB04] C. Pelsser, S. Uhlig, and O. Bonaventure. On the difficulty of establishing interdomain LSPs. In *IEEE International Workshop on IP Operations and Management (IPOM 2004)*, Beijing, China, October 11-13th 2004.

[QB05] B. Quoitin and O. Bonaventure. A cooperative approach to interdomain traffic engineering. In *1st Conference on Next Generation Internet Networks Traffic Engineering (NGI 2005)*, Rome, Italy, April 18-20th 2005.

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

[QU05] B. Quoitin and S. Uhlig. Modeling the routing of an Autonomous System with C-BGP. *IEEE Network*, 19(6), November 2005.

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

[RD04] F. Ricciato and A. D'Achille. A novel scheme for end-to-end protection in a multi-area network. In *Inter-Domain Performance and Simulation (IPS 2004)*, Budapest, Hungary, 22-23 March 2004.

[RIP06] IPv4 address allocation and assignment policies for the ripe ncc service region. `http://www.ripe.net/ripe/docs/ipv4-policies.html`, February 2006.

[RLH06] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, January 2006.

[Rou] University of oregon route views project. http://www.routeviews.org.

[RR01] Y. Rekhter and E. Rosen. Carrying label information in BGP-4. RFC 3107, May 2001.

[RR06] E. Rosen and Y. Rekhter. BGP/MPLS IP Virtual Private Networks (VPN)s, February 2006. RFC 4346.

[RV05] R.Zhang and J.-P. Vasseur. MPLS inter-Autonomous System (AS) Traffic Engineering (TE) requirements. RFC 4216, November 2005.

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

[SB06] M. Shand and S. Bryant. IP fast reroute framework. Internet draft, draft-ietf-rtgwg-ipfrr-framework-06.txt, work in progress, October 2006.

[SG04] Aman Shaikh and Albert Greenberg. OSPF Monitoring: Architecture, Design and Deployment Experience. In *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, March 2004.

[SH03] V. Sharma and F. Hellstrand. Framework for Multi Protocol Label Switching (MPLS)-based Recovery. RFC 3469, February 2003.

[SL04] H. Smit and T. Li. Intermediate System to Intermediate System (IS-IS) extensions for Traffic Engineering (TE). RFC 3784, June 2004.

[SMWA04] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, February 2004.

[SOIU06] K. Shiomoto, E. Oki, I. Inoue, and S. Urushidani. A server-based traffic engineering method in IP+Optical multi-layer networks. In *Proc. of the International Conference on IP+Optical Networks (iPOP 2006)*, Tokyo, Japan, June 22nd-23rd 2006.

[SPL$^+$06] K. Shiomoto, D. Papadimitriou, J-L. Leroux, M. Vigoureux, and D. Brungard. Requirements for GMPLS-based multi-region and multi-layer networks (MRN/MLN). Internet draft, draft-ietf-ccamp-gmpls-mln-reqs-01.txt, work in progress, June 2006.

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

[TGRR05] R. Teixeira, T. Griffin, M. G. C. Resende, and J. Rexford. Tie breaking: Tunable interdomain egress selection. In *Proc. of CoNext'2005*, October 2005.

[TH00] D. Thaler and C. Hopps. Multipath issues in unicast and multicast next-hop selection. RFC 2991, Nvember 2000.

[TJX$^+$06] P. Torab, B. Jabbari, Q. Xu, S. Gong, X. Yang, T. Lehman, C. Tracy, and J. Sobieski. On cooperative inter-domain path computation. In *11th IEEE Symposium on Computers and Communications (ISCC'06)*, pages 511–518, June 2006.

[TMSV03] R. Teixeira, K. Marzullo, S. Savage, and G. M. Voelker. In search of path diversity in ISP networks. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 313–318, New York, NY, USA, 2003. ACM Press.

[Tra96] P. Traina. Autonomous system confederations for BGP. Internet RFC 1965, June 1996.

[UBQ03] S. Uhlig, O. Bonaventure, and B. Quoitin. Interdomain traffic engineering with minimal BGP configurations. In *18th International Teletraffic Congress (ITC)*, September 2003.

[UQ05] S. Uhlig and B. Quoitin. Tweak-it: BGP-based interdomain traffic engineering for transit ASes. In *1st Conference on Next Generation Internet Networks Traffic Engineering (NGI 2005)*, Rome, Italy, April 18-20th 2005.

[UT06] S. Uhlig and S. Tandel. Quantifying the impact of route-reflection on BGP routes diversity inside a tier-1 network. In *Proceedings of Networking 2006*, Coimbra, Portugal, May 15-19th 2006.

[VAS06] J-P. Vasseur, Z. Ali, and S. Sivabalan. Definition of a Record Route Object RRO node-id sub-object. RFC 4561, June 2006.

[VAZ06] J-P. Vasseur, A. Ayyangar, and R. Zhang. A per-domain path computation method for establishing inter-domain traffic engineering (TE) label switched path (LSP). Internet draft, draft-ietf-ccamp-inter-domain-pd-path-comp-03.txt, work in progress, August 2006.

[VIZ06] J-P. Vasseur, Y. Ikejiri, and R. Zhang. Reoptimization of MultiProtocol Label Switching (MPLS) Traffic Engineering (TE) loosely routed Label Switch Path (LSP). Internet draft, draft-ietf-ccamp-loose-path-reopt-02.txt, work in progress, February 2006.

[VLA$^+$06] J-P. Vasseur, J-L. Le Roux, A. Ayyangar, E. Oki, A. Atlas, A. Dolganow, Y. Ikejiri, and K. Kumaki. Path computation element (PCE) communication protocol (PCEP) - version 1. Internet draft, draft-ietf-pce-pcep-02.txt, work in progress, June 2006.

[VPD04] J-P. Vasseur, M. Pickavet, and P. Demeester. *Network Recovery - Protection and Restoration of Optical, SONET-SDH, IP and MPLS*. Morgan Kauffmann, 2004.

[VVKB06] Mythili Vutukuru, Paul Valiant, Swastik Kopparty, and Hari Balakrishnan. How to Construct a Correct and Scalable iBGP Configuration. In *IEEE INFOCOM*, Barcelona, Spain, April 2006.

[VZB06] J-P. Vasseur, R. Zhang, and N. Bitar. A Backward Recursive PCE-based Computation (BRPC) procedure to compute shortest inter-domain Traffic Engineering Label Switched Path. Internet draft, draft-vasseur-ccamp-brpc-00.txt, work in progress, February 2006.

[WCCL05] H. Wang, R. K. C. Chang, D-M. Chiu, and J. C. S. Lui. Characterizing the performance and stability issues of the AS Path prepending method: Taxonomy, measurement study and analysis. In *Proc. of ACM SIGCOMM Asia Workshop*, April 2005.

[XHBN00] X. Xiao, A. Hannan, B. Bailey, and L. Ni. Traffic engineering with MPLS in the Internet. *IEEE Network Magazine*, pages 28–33, March 2000.

[XLWN02] L. Xiao, K.-S. Lui, J. Wang, and K. Nahrstedt. QoS extension to BGP. In *10th IEEE International Conference on Network Protocols*, Paris, France, November 12-15 2002.

[YLT$^+$06] X. Yang, T. Lehman, T. Tracy, J. Sobieski, P. Torab, S. Gong, and B. Jabbari. Policy-based resource management and service provisioning in GMPLS networks. In *First IEEE Workshop on Adaptive Policy-based Management in Network Management and Control*, Barcelona, Spain, April 2006.

[YSLMB$^+$05] M. Yannuzzi, S. Sánchez-López, X. Masip-Bruin, J. Solé-Pareta, and J. Domingo-Pascual. A combined intra-domain and inter-domain qos routing model for optical networks. In *9th conference on Optical Network Design and Modelling (ONDM 2005)*, Milan, Italy, February 7-9th 2005.

[ZCD97] Ellen W. Zegura, Kenneth L. Calvert, and Michael J. Donahoo. A quantitative comparison of graph-based models for Internet topology. *IEEE/ACM Transactions on Networking*, 5(6):770–783, 1997.

[ZK05] Shu Zhang and Katsushi Kobayashi. Rtanaly: A system to detect and measure igp routing changes. In *IPOM*, pages 162–172, 2005.