UNIVERSITÉ DE LIÈGE
Faculté des Sciences Appliquées
Département d'Électricité,
Électronique et Informatique

Contributions to Traffic Engineering and Resilience in Computer Networks

Thèse présentée par

Simon Balon

en vue de l'obtention du titre
de Docteur en Sciences de l'Ingénieur

Année académique 2008–2009

Abstract

The Internet traffic is constantly increasing following the emergence of new network applications like social networks, peer-to-peer, IP phone or IP television. In addition, these new applications request better path availability and path quality. Indeed the efficiency of these applications is strongly related to the quality of the underlying network. In that context network operators make use of traffic engineering techniques in order to improve the quality of the routes inside their network, but also to reduce the network cost of increased traffic handling with a better utilization of existing resources. This PhD thesis covers several topics of Traffic Engineering and Fast Restoration in IP/MPLS networks.

Our first contribution is related to the definition of a well-engineered network. In the literature mathematical formulation of Traffic Engineering (TE) requirements are very diverse. We have thus performed a comparative study of many objective functions, in order to differentiate them and choose in a rational way the one that best reflects Traffic Engineering goals. We have also designed a method approaching optimal TE, whereby we divide the traffic matrix in N sub-matrices and route them independently, based on the derivatives of the objective function.

The second topic addressed in this work concerns link weight optimizers (LWOs). Link weight optimization is the traffic engineering *"standard"* technique in networks running link state routing protocols (which are widely used in transit networks). These link weight optimizers suffer from several limitations due to the BGP (Border Gateway Protocol) Hot-Potato rule, which is basically not considered by such optimizers. Therefore we have proposed a BGP-aware link weight optimization method that takes problematic Hot-Potato effects into account, and even turns them into an advantage. We have also studied how LWOs behave in big networks which have to use BGP route reflectors. Finally we have studied whether forwarding loops can appear or not when traffic is split among multiple equivalent egress routers, an optional BGP feature that we did use in our Hot-Potato aware LWO.

Our last contribution concerns network resilience. We have proposed a solution for a rapid recovery from a link or node failure in an MPLS network. Our solution allows a decentralized deployment combined with a minimal bandwidth usage while requiring only reduced amount of information to flood in the network. This method is the first that makes possible a decentralized deployment combined with an optimal resource consumption.

To easily simulate and test the methods proposed in this work, we have also contributed to the development of TOTEM - a TOolbox for Traffic Engineering Methods.

Acknowledgments
Remerciements

I want to sincerely acknowledge the members of my thesis committee, namely Olivier Bonaventure, Raouf Boutaba, Bernard Fortz, Roch Guérin, Guy Leduc and Pierre Wolper, for their attentive review and their interest in my research work.

(in french)

Tout d'abord, je tiens à remercier sincèrement le Professeur Guy Leduc qui a accepté d'être le promoteur de ce travail. Les discussions nombreuses avec lui, ses conseils, mais aussi son soutien ont été très précieux. Il m'a également permis de me concentrer sur mes travaux sans me surcharger de tâches administratives.

Merci aux membres de l'unité de recherche en réseaux informatiques : Bamba, Charline, Cyril, Cyrine, Dali, Fabian, François, Gaël, Ibtissam, Jean, Jean-Marc, Laurent, Nicolas, Olivier, Sandrine et Sylvain. Ils m'ont souvent fourni leurs conseils avisés et permis de travailler dans une ambiance de travail agréable et stimulante. Merci aux membres de l'UCL pour les nombreuses discussions et conseils : Olivier Bonaventure, Steve Uhlig, Bruno Quoitin, Cristel Pelsser, Pierre François, Sébastien Tandel, Virginie Van den Schrieck, Bernard Fortz, Hakan Umit et Selin Cerav-Erbas.

Plus globalement, merci à toutes les personnes que j'ai rencontrées lors de ces années à l'Université.

Merci également aux institutions suivantes, différentes sources de financement de ce travail : le FNRS, la région Wallonne et l'Union Européenne.

Pour finir, merci à Hélène Libert, ma compagne et à mon fils Alexandre pour leur soutien et leur réconfort quotidien. Merci également à ma famille et plus particulièrement mes parents.

Contents

List of Figures

1

Introduction to Traffic Engineering (TE) in Computer Networks

Today the Internet is used as a support for a significant part of the world-wide communications. The development of efficient peer-to-peer systems, cheap phone call softwares, but also the Internet television will probably increase this phenomenon in the near future. But the Internet was first designed as a best-effort network without any guarantee. With the development of new applications running on top of the Internet, it has to deal with increasing traffic load, but also with users that more and more request better path quality in terms of bandwidth and delays, and also for a better network availability. During the past years increased quality of service was implemented by overprovisioning the link capacities, but it will be less and less feasible for network operators in the future, as fast as the traffic grows. In that context it is important to use Traffic Engineering techniques to make a better use of the network resources, or more generally *to put the traffic where the network bandwidth is available*, in every possible situation.

Traffic Engineering deals with optimizing the network routes in order to better drive the traffic through the network, avoiding overloaded links. Said in another way, Traffic Engineering techniques have to optimally map the traffic to the underlying topology. Optimizing the network routes first requires to understand how the different routing protocols interact to determine the end-to-end paths of traffic in the Internet. The Internet is composed of the interconnection of thousands of autonomous systems which are independently managed.

We will describe Internet routing in section 1.1. Then section 1.2 introduces Traffic Engineering concepts and section 1.3 describes our contributions. Finally section 1.4 presents the publications related to this work.

1.1 Routing in the Internet

Each packet sent on the Internet follows a path which is defined by routing protocols. The exterior gateway protocol (EGP) defines the path at the network-

level. This path is called the AS path[1], which is basically composed of the succession of every AS on the way from a source host to a destination host. The EGP used in the Internet is BGP (Border Gateway Protocol). In each AS the path from each ingress router to each egress router is defined by the interior gateway protocol (IGP). There are two types of IGPs: link state versus distance vector. The IGPs most commonly used in transit networks are OSPF ([Moy98]) and ISIS ([Cal90]), which are both link state routing protocols. Recently MPLS has also been introduced in many ASes. The path followed by a packet is the combination of both the IGP and the EGP. The topology of figure 1.1 will be used to illustrate some concepts introduced in this section.

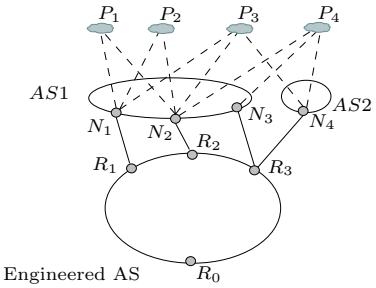

Figure 1.1: Example Topology

1.1.1 Intradomain Routing Protocol: IGPs and MPLS

Routers that are part of a domain running a link state routing protocol periodically exchange link state information. Each router in the network floods it with link state packets containing information about its directly connected links. For example, if a router is connected to three links, it will send to every other router of the domain information about these three links. Basically the information about the link is the neighbor router (the router at the other end of the link) and an administrative weight associated with the link by a network administrator. So each router receives link state packets from every other router of the domain. Then each router can combine all this information to build a complete view of the AS topology.

In an AS the path between ingress and egress routers are computed by a Shortest-Path algorithm based on the link weights. All the traffic is sent on the shortest path (the path whose sum of link weights is minimum). If ECMP (Equal Cost Multi-Path) is enabled, several equal minimum cost paths can be used simultaneously to evenly split the traffic among them. In that case some routers use a simple round-robin mechanism to equally balance the load on multiple next-hops. But a better solution is that routers use a hash table that maps a hash of multiple fields in the packet header to one of these paths, so that every packet of a flow will follow the same path with limited packet reordering (see [CWZ00] for a performance analysis of hashing based schemes for Internet load balancing). This means that in practice it may not be a perfect even split of traffic on every shortest path. Some recent work proposes to split traffic with

[1]AS stands for Autonomous System. In the work we use domain and AS interchangeably.

a finer granularity based on *flowlet* ([SKK04]). Figure 1.2 shows an example of ECMP inside an AS. This figure assumes that there are two equal cost paths from R_0 to R_1.

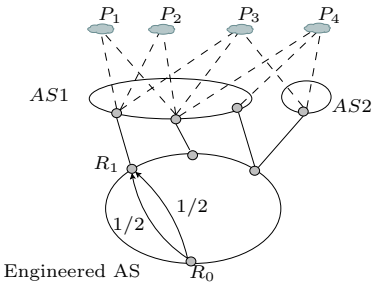

Figure 1.2: Intradomain Equal Cost Multipath (ECMP)

More routing flexibility is provided by MPLS (Multi-Protocol Label Switching). MPLS allows the network operator to establish explicit tunnels between routers. In the MPLS context tunnels are called Label Switched Paths (LSP). A classical MPLS configuration inside an AS is a full-mesh, where one tunnel is established from every ingress router to every egress router. With this configuration, when an IP packet enters the MPLS network, the first MPLS router that handles it (i.e. the ingress router) encapsulates it in an MPLS packet and forwards it on the LSP whose end router is the egress for that packet. Inside the MPLS network the packet is forwarded based on its MPLS header only. The end router of the LSP decapsulates the packet which is then routed like a classical IP packet. If multiple MPLS LSP tunnels are available between an ingress and an egress router, it is possible to specify an arbitrary split of traffic between these tunnels.

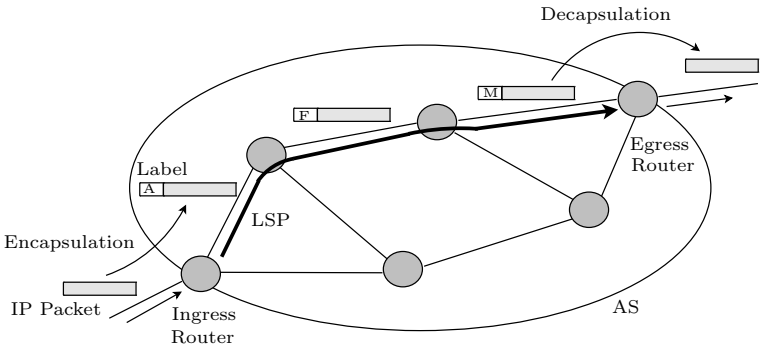

Figure 1.3: MPLS Routing

MPLS forwarding is based on fixed-length labels which are contained in the MPLS headers, and not on the IP destination address. The labels are defined at LSP setup. Label Distribution Protocol (LDP) is defined in [ADF+01] for distribution of labels inside one MPLS domain. The setup of constraint-based LSPs using LDP is described in [JAC+02]. An extension of RSVP can also be used to establish LSPs ([ABG+01]). Each MPLS router has a routing table which associates with each (input interface, input label) pair an (output

interface, output label) pair.

Note that the concept of Traffic Engineering was first introduced in MPLS networks ([AMA+99, Awd99, XHBN00, ACE+02, AJ02]). The advantages of MPLS for Traffic Engineering are the capacity of explicit routing and arbitrary splitting of traffic. Each LSP tunnel can be routed independently of each other. This is of course not the case when shortest paths are used as in that case all the paths are dependent of the same common set of link weights.

1.1.2 Interdomain Routing Protocol: BGP

BGP allows routers to exchange reachability information between neighboring ASes ([Ste99]). Each AS is connected to several neighboring ASes by interdomain links. Depending on the connectivity of the network and on the destination of the packet, one or several neighboring ASes can be chosen to forward the packet to the destination. The choice of the BGP next-hop (i.e. the egress router in this AS or the border router in the next AS, that will relay the packet toward the destination) is based on the information exchanged with neighbors and on a local configuration implementing its routing policy.

There are two types of BGP sessions that are used to exchange routes between routers. eBGP sessions are used between routers in different ASes, while iBGP sessions are used between routers in the same AS. When a router receives a route on a iBGP or eBGP session, this route has to pass an input filter to be eligible in the BGP decision process. This process selects the best route toward each destination IP prefix[2]. The best route selected by this process is then forwarded on other BGP sessions after passing through an output filter. The whole process is described on figure 1.4.

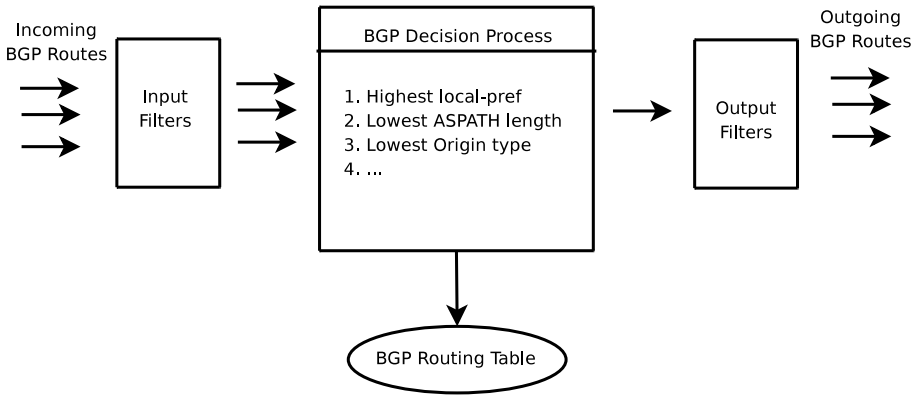

Figure 1.4: BGP route selection

The BGP route selection process (the central box of figure 1.4), implementing routing policies, is made of several criteria ([BGP, Fou]):

1) Prefer routes with the highest local preference which reflects the routing policies of the domain;

[2]An IP prefix is a block of IP addresses, i.e. a set of end host's Internet addresses.

2) Prefer routes with the shortest AS-level Path;

3) Prefer routes with the lowest origin number, e.g., the routes originating from IGP are most reliable;

4) Prefer routes with the lowest MED (multiple-exit discriminator) type which is an attribute used to compare routes with the same next AS-hop;

5) Prefer eBGP-learned routes over iBGP-learned ones (referred to as the eBGP>iBGP criterion in this work);

6) Prefer the route with the lowest IGP distance to the egress point (i.e. the so-called hot-potato, or early exit, criterion);

7) If supported, apply load sharing between paths. Otherwise, apply a domain-dependent tie-breaking rule, e.g., select the one with the lowest egress ID.

We will illustrate how BGP is running on the network of figure 1.5. P_1, P_2, P_3 and P_4 are four IP destination prefixes. Dashed lines represent an available BGP inter-AS route. So N_1 and N_2 have a path to prefix P_1 and P_2; N_1, N_2 and N_4 have a path to prefix P_3, ... Now suppose that these routes pass through the output filters of these routers and that these are forwarded on eBGP sessions from N_1 to R_1, from N_2 to R_2, ... Then R_1, R_2 and R_3 receive the BGP routes that have to pass through input filters. These are then used in the BGP decision process and are selected as best routes as these are the only available routes for the moment. Then the selected routes are forwarded on iBGP sessions inside the AS. A classical iBGP configuration inside an AS is an iBGP full-mesh where every router has an iBGP session with every other BGP router in the AS. With this configuration every BGP router receives on an iBGP session the best route chosen by every other BGP router in the AS. If there is an iBGP full-mesh in the network we consider, R_0, R_1, R_2 and R_3 will receive every available route.

If we now consider router R_0, it has multiple available routes toward every prefix. It will use the BGP route selection process to choose the best one among all available ones. For example on figure 1.5 we see that R_0 has to choose between three available routes to prefix P_3.

In this example we did suppose that an iBGP full-mesh was configured inside the AS. In big networks operators use route reflectors ([BCC00]) to decrease the number of iBGP sessions inside the AS compared to a full-mesh configuration.

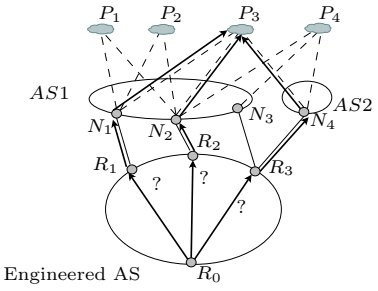

Figure 1.5: Three available BGP routes

1.2 Traffic Engineering

As we have seen in section 1.1, the end-to-end paths used in the Internet result from the combination of both Inter- and Intra-domain routing protocols. Traffic Engineering methods tune the parameters of these protocols to optimize the paths of traffic.

We can define Traffic Engineering as follows. *Traffic engineering involves adapting the routing of traffic to the network conditions, with the joint goals of good user performance and efficient use of network resources.*

Usually intradomain TE has different objectives than interdomain TE. Intradomain Traffic Engineering deals with the optimization of routes inside the network. The goal of this kind of optimization is for example to equally balance the load on intradomain links, to avoid bottleneck links, to minimize congestion or to minimize path delays. On the other side Interdomain Traffic Engineering considers the tuning of interdomain routing protocol parameters to impact the inter-domain paths. The goal is to balance the traffic on interdomain links, by choosing the ingress and/or the egress routers for some part of the traffic.

1.2.1 Intradomain Traffic Engineering

Consider a network running a classical link state intradomain IGP protocol (ISIS or OSPF). The routes in the network will be computed by a Shortest Path First (SPF) algorithm based on some link weights assigned by the network administrator. By default, the weights can either be all set to 1 (leading to a minimum hop routing) or to the inverse of the capacity of the links (as recommended by CISCO), for example.

We will see in chapter 5 that these simple routing configurations can be proven to optimize some simple traffic engineering objectives. But these routing configurations do not take traffic into account and thus can lead to some problems. Indeed, one link can be highly loaded (many flows are routed via this link) while others are nearly not used. The high load of some links can lead to congestion or to a high delay due to packet queuing. In this case, it is known that the quality of the network would be improved if some flows were routed on a somewhat longer but less loaded (i.e. with more available bandwidth) path. One high level objective of a simple traffic engineering technique could be to balance the load over all links by trying to decrease the load of the most loaded link(s).

We will illustrate these concepts on the network of figure 1.6. Suppose that each link has a weight of 1. Traffic goes from node S_1 to D and from node S_2 to D. The cost of path $S_1 \rightarrow N_1 \rightarrow N_2 \rightarrow D$ is equal to the cost of path $S_2 \rightarrow N_1 \rightarrow N_2 \rightarrow D = 3 < 4 =$ the cost of path $S_1 \rightarrow N_1 \rightarrow N_3 \rightarrow N_4 \rightarrow D$ which is equal to the cost of path $S_2 \rightarrow N_1 \rightarrow N_3 \rightarrow N_4 \rightarrow D$. This implies that both flows will use the upper path which may be congested while the bottom path is still underutilized.

A Traffic Engineering method should try to optimize the routes to balance the load on both paths and avoid traffic congestion on the upper path. After Traffic Engineering, the network routes should be as depicted on figure 1.7. This

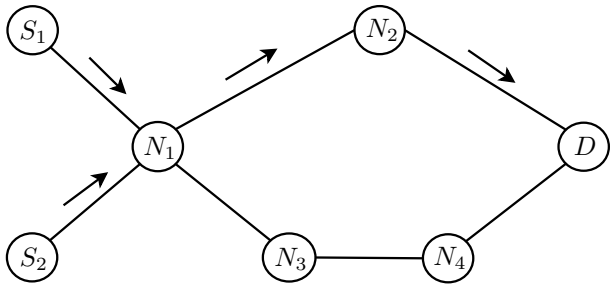

Figure 1.6: Fish topology: hop count shortest paths

routing scheme provides a better usage of network resources and also better user performance (as congestion is avoided).

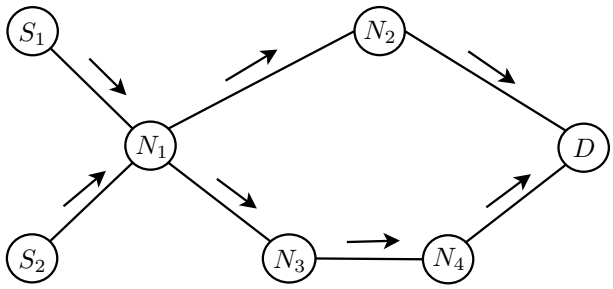

Figure 1.7: Fish topology: Split the traffic on both paths

This simple example highlights that Traffic Engineering techniques should take traffic into account to compute the traffic routes.

Typically, off-line traffic engineering methods optimize the routes based on aggregated traffic information: the Traffic Matrix. The Traffic Matrix (TM) associates with each pair of (ingress, egress) routers inside the AS a traffic value. This can for example be the mean traffic value that has been measured during the peak hour. Optimizers can use multiple Traffic Matrices as input, each TM being associated with one particular network state. We explain how to obtain a Traffic Matrix in chapter 3.

1.2.1.1 Traffic Engineering in OSPF/ISIS networks

The first technique that can achieve the traffic engineering objective as defined in the preceding section is the following. Find a set of link weights such that when the shortest paths will be computed with respect to these weights, the load will be balanced on the whole network and the maximum link load is minimized. This problem of finding the set of weights that minimizes the load of the most loaded link(s) is combinatorial and some heuristics to solve it have been proposed in [FT00, FRT02]. We detail the different versions of LWOs in chapter 2.

1.2.1.2 Traffic Engineering in MPLS networks

A completely different solution to this complex problem is to factorize it into several simpler ones. In an MPLS network, the paths (and the granularity) of all the flow aggregates of the network can be chosen freely. With this kind of tunnel-based technology, it is possible to route all the flows with the goal of optimizing one specific objective function (or a combination of several ones). If the traffic matrix changes, it is possible to reroute or reoptimize only some of the LSPs, while avoiding classical transient loop problems. To recover from failures, it is possible to precompute and pre-establish some backup LSPs. One backup LSP will only be active when the corresponding primary LSP has failed [MCSA03]. Again, the paths of these backup LSPs can be freely chosen so that in case of any failure no congestion will occur.

MPLS routing is thus somewhat more complicated than pure IP routing, but it allows more flexibility and more degrees of freedom than IP's shortest path routing. It is also possible to use hybrid solutions combining shortest path routing for most flows and MPLS tunnels for some traffic aggregates. The essential merit of this hybrid approach is to avoid a full mesh of LSPs, which may be impractical for very large networks ([SBL06]).

1.2.2 Interdomain Traffic Engineering

The goal of Interdomain TE is to control incoming and/or outgoing traffic. There are multiple motivations for interdomain traffic engineering. In [FBR03] the authors provide three examples to motivate the need for interdomain TE :

- Congested inter-domain link. If a network operator detects that an interdomain link is congested, it can change the interdomain paths to divert some traffic from that link.
- Upgraded link capacity. When the capacity of an interdomain link is upgraded, the network operator wants to drive more traffic on that link.
- Violation of peering agreement. Some ASes have commercial agreements that limit the amount of traffic they exchange. If the limit is exceeded the network operator may need to reduce the traffic sent on corresponding interdomain links, which can be done by changing the route for part of the traffic flowing on these links.

Interdomain TE is not a trivial task for several reasons. The main reason is that a network operator has only an indirect and incomplete control on the interdomain paths. BGP routing policies are flexible but complex. The best route chosen by a router depends on these policies, but also on the BGP routes received by neighboring ASes. The system is distributed and a best route change in one AS can result in the best route change in one of its neighboring AS. As a result it is very difficult to predict the total effect on the global system of a local change in one particular AS.

Interdomain TE is generally performed in a trial-and-error manner ([UQ05]). The different techniques that can be used to control the traffic on interdomain links are described in [QUP+03].

1.3 Contributions of this work

Our first contribution is related to the definition of a well-engineered network. In the literature mathematical formulation of Traffic Engineering (TE) requirements are very diverse. We have thus performed a comparative study of many objective functions, in order to differentiate them and choose in a rational way the one that best reflects Traffic Engineering goals. We have also designed a method approaching optimal TE, whereby we divide the traffic matrix in N submatrices and route them independently, based on the derivatives of the objective function.

The second topic addressed in this work concerns link weight optimizers (LWOs). Link weight optimization is the traffic engineering *"standard"* technique in networks running link state routing protocols (which are used in most transit networks). These link weight optimizers suffer from several limitations due to the BGP (Border Gateway Protocol) Hot-Potato rule, which is basically not considered by such optimizers. Therefore we have proposed a BGP-aware link weight optimization method that takes problematic Hot-Potato effects into account, and even turns them into an advantage. We have also studied how LWOs behave in big networks which have to use BGP route reflectors. Finally we have studied whether forwarding loops can appear or not when traffic is split among multiple equivalent egress routers, an optional BGP feature that we did use in our Hot-Potato aware LWO.

Our last contribution concerns network resilience. We have proposed a solution for a rapid recovery from a link or node failure in an MPLS network. Our solution allows a decentralized deployment combined with a minimal bandwidth usage while requiring only reduced amount of information to flood in the network. This method is the first that makes possible a decentralized deployment combined with an optimal resource consumption.

To easily simulate and test the methods proposed in this work, we have also contributed to the development of TOTEM - a TOolbox for Traffic Engineering Methods.

1.3.1 Definition of a Well-Engineered Network

Basically engineering the traffic of a computer network can be seen as solving an optimization problem: the network operator has to find the *"best"* routing scheme for current traffic and network conditions. The traffic and network conditions determine constraints that must be respected while the operator can tune routing parameters that determine the traffic routes. A fundamental but not well understood problem in Traffic Engineering is the mathematical formulation of the best routing scheme, which reflects the Traffic Engineering goals and objectives. In the literature we can find a series of requirements that a well-engineered network should fulfill, such as: minimize the delay, minimize losses, avoid having some parts of the network over-utilized while other parts are under-utilized, ... All these requirements have been published, for example by the IETF (Internet Engineering Task Force). We have noticed that it is very difficult to obtain a mathematical formulation of all these requirements. This is probably why many scientific papers propose some mathematical functions to

evaluate the quality of a solution (i.e. a network state) based on precited requirements, but proposed functions never include all these requirements. In that context it is difficult to figure out which function is the best. **We have thus performed an extensive study of several objective functions found in the literature, in order to differentiate them and determine a rational way to choose one function that best reflects Traffic Engineering goals ([BSL05, BSL06]).**

A good TE objective function reflecting real needs of network operators is required in every TE algorithm. Usually TE problems are combinatorial and efficient heuristics are needed to find near-optimal solution in reasonable time. In that context **we have developed a new method to find a routing scheme that approaches the optimum defined by the objective function ([BL06]).** This method divides the traffic matrix in N sub-matrices and routes each of these independently. By default we use the best objective function determined in previous study but we can also use another one. We outline two possibilities to implement this method in routers.

1.3.2 Link Weight Optimizers and BGP Hot-Potato rule

We have also studied several limitations of the classical and quite well-studied method to perform TE inside an AS running a link state routing protocol (OSPF/ISIS). This classical method consists in using a link weight optimizer (LWO) to try and find the optimal set of link weights for a given intradomain traffic matrix. We have proposed several contributions and improvements to the state of the art of LWOs.

The first limitation is the following: it has been shown that the classical approach is inadequate because it ignores a potential impact on interdomain routing. Indeed, the resulting set of link weights may trigger BGP to change the BGP next hop for some destination prefixes, to enforce hot-potato routing policies. In turn, this results in changes in the intradomain traffic matrix that have not been anticipated by the link weight optimizer, possibly leading to degraded network performance. We have first quantified this degraded performance on a real dataset, on which the worst case traffic matrix has led to a maximal link utilization over 100%, while the optimizer predicted less than 30%. Then **we have proposed a new BGP-aware link weight optimization method that takes hot-potato effects into account, and even turns them into an advantage ([BL08]).** This method uses the interdomain traffic matrix and other available BGP data, to extend the intradomain topology with external virtual nodes and links, on which all the well-tuned heuristics of a classical link weights optimizer can be applied. **A key innovative asset of our method is its ability to also optimize the traffic on the interdomain peering links.** We have shown that our approach does so efficiently at almost no extra computational cost, by testing it on a real dataset.

We have then found another limitation of LWOs. While the second generation of LWOs has been able to optimize link weights while taking hot-potato effects into account, these tools relied on the complete visibility assumption fulfilled by e.g. a full-mesh iBGP configuration. In iBGP configurations based on route reflectors, which usually hide some reachability information from routers,

this second generation of LWOs suffers from some problems, caused by the partial visibility, including path deflections (i.e., the actual egress router is not the expected one), which may in turn create forwarding loops. **We have proposed a (third generation) LWO which embeds a BGP routing solver that can always predict the actual egress router, even when route reflectors are used. It can also forbid solutions leading to path deflection.** Its efficiency has also been evaluated on a real dataset.

Finally we did study a last problem related to BGP-aware LWOs. These LWOs can benefit from iBGP multipath load sharing to balance the traffic on multiple inter-domain links, as do classical LWOs with Equal Cost Multipath to balance the load on multiple paths inside the AS. But it has been said that splitting the traffic amongst multiple available BGP-equivalent routes can lead to forwarding loops. **We have shown that under reasonable assumptions, forwarding loops should not appear when iBGP multipath load sharing is used.** Indeed we have shown that BGP configurations reflecting commercial relationships ensure that no forwarding loops are created. Anyway as it is not possible for a network operator to verify the configurations of all the involved ASes in the Internet, we have analyzed what would happen if BGP configuration would not reflect commercial relationships, even though in principle it should be the case. We have shown that even in this case, a forwarding loop cannot appear immediately after activating iBGP multipath. The forwarding loop could only appear if in addition to the aforementioned conditions, some ASes change their policies in a particular way. Moreover we have shown that even in this case, if a forwarding loop appears, it is only transient. This leads us to conclude that activating iBGP multipath for routes with different ASPATH is not as dangerous as it may seem at first.

1.3.3 Resilience in MPLS networks

Finally **we have studied a peripheral TE problem related to resilience in case of network failures.** Inside an AS, when a link or node fails, the intra-domain routing protocol (OSPF or ISIS) needs some time to converge to a new stable routing state. During this time period, all the packets that were initially routed via this link or node are lost when multiple paths are not available. Moreover once the routing protocol has converged, it is not sure that there is no congestion in the network. In MPLS networks it is possible to establish in advance backup paths that will be used in case of failure. Thanks to these preestablished paths, it is possible to reduce the duration of service unavailability to the minimum. These backup paths can also be computed so that no congestion will appear in the network after the failure. But this protection has a cost in terms of bandwidth consumption. Indeed for an optimal protection, bandwidth has to be reserved for backup paths. In our research unit Mélon et al. had previously proposed a sophisticated resource aggregation technique based on the concept of bandwidth sharing between backup paths, but also between primary and backup paths ([MBL03]). To deploy this algorithm in a decentralized environment, each node which computes a backup path needs some information to optimise this computation. This information is the bandwidth reserved on each link of the network for primary paths, but also for backup paths. The computing node also needs to know when bandwidth reserved for

backup will be used (i.e. we have to specify which link and/or node failures will result in activating backup paths using this bandwidth). This means that a huge amount of data may flow through the network. Indeed a naive approach is to flood the network (i.e. one message on each link of the network) with a message each time the information related to the reserved bandwidth on a link changes (for example because a path has been established on this link). To solve this problem **we have proposed a solution where the nodes obtain almost all the information they need from RSVP messages** (RSVP is used to establish paths in the network). **This mechanism drastically reduces the amount of data to flood in the network.** This method is the **first which makes possible a decentralized deployment combined with an optimal resource consumption ([BML06]).**

1.3.4 The TOTEM Toolbox

We also want to note our contribution to the **development of TOTEM, a TOolbox for Traffic Engineering Methods**. The goal of this toolbox was to develop an open source software collecting a great variety of traffic engineering (TE) algorithms. Once integrated in the toolbox, these different TE algorithms can be combined for a particular purpose, which is one of the key advantages of such a toolbox. The utility of the toolbox is much higher than the sum of the utilities of the embedded algorithms. We have integrated MPLS and pure IP Intradomain TE tools but also BGP Interdomain TE tools. **We have demonstrated the usefulness of the TOTEM toolbox by testing its capabilities on a real dataset.** In that case study we have analyzed and compared different traffic engineering methods, which highlighted the pros and cons of each method ([SBD+05, BLD+07]). Note also that **all the methods presented in this thesis have been implemented and tested in the TOTEM toolbox framework.**

A frequent problem for researchers that want to asses the quality of a new method is the difficulty to access to real operational data concerning topologies and traffic matrices. It is important to simulate new traffic engineering algorithms on real and not only on randomly generated data as this gives more credibility to the study. It is then easier to convince reviewers of the real efficiency of the new algorithm. We have also collaborated with UCL to **obtain real traffic measurements ([UQLB06]).**

1.4 Publications

[BL09] (In Chapter 7) - S. Balon and G. Leduc. BGP-aware IGP Link Weight Optimization in Presence of Route Reflectors. Proceedings of IEEE INFOCOM, 20-24 Apr. 2009, Rio de Janeiro, Brazil (Acceptance ratio = 282/1435 = 20%)

[BL08] (In Chapter 6) - S. Balon and G. Leduc. Combined Intra- and Interdomain Traffic Engineering using Hot-Potato Aware Link Weights Optimization. *SIGMETRICS Perform. Eval. Rev.*, vol.36, No.1, pp 441-442, 2008
Proc. of ACM SIGMETRICS 2008 - International Conference on Measurement

and Modeling of Computer Systems, pp. 441-442, 2-6 Jun. 2008, Annapolis, Maryland, USA.

[BL07] (In Chapter 8) - S. Balon and G. Leduc. Can Forwarding Loops Appear when Activating iBGP Multipath Load Sharing ? *Proceedings of the Third Asian Internet Engineering Conference (AINTEC 2007)*, 27-29 Nov. 2007, Phuket, Thailand, S. Fdida and K. Sugiura (ed.), Sustainable Internet, LNCS 4866, pp. 213-225, Springer-Verlag (Acceptance ratio = 14/66 = 21%).

[BLD+07] (In Chapter 3) - S. Balon, J. Lepropre, O. Delcourt, F. Skivée and G. Leduc. Traffic Engineering an Operational Network with the TOTEM Toolbox. *IEEE Transactions on Network and Service Management*, Vol.4 No.1, pp 51-61, June 2007.

[BL06] (In Chapter 5) - S. Balon and G. Leduc. Dividing the Traffic Matrix to Approach Optimal Traffic Engineering. *IEEE International Conference on Networks (ICON)*, vol.2, pp. 566-571, 13-15 Sep. 2006, Singapore (Acceptance ratio = 101/261 = 38%).

[SBL06] - F. Skivée, S. Balon and G. Leduc. A scalable heuristic for hybrid IGP/MPLS traffic engineering - Case study on an operational network. *IEEE International Conference on Networks (ICON)*, vol.2, pp. 572-577, 13-15 Sep. 2006, Singapore (Acceptance ratio = 101/261 = 38%).

[BML06] (In Chapter 9) - S. Balon, L. Mélon and G. Leduc. A scalable and decentralized fast-rerouting scheme with efficient bandwidth sharing. *Computer Networks*, vol. 50, nb. 16, Nov. 2006, pp. 3043-3063. (ISI IF 2007 = 0.829).

[BSL06] (In Chapter 4) - S. Balon, F. Skivée and G. Leduc. How Well Do Traffic Engineering Objective Functions Meet TE Requirements? *Proc. of IFIP International Networking Conference*, 15-19 May 2006, Coimbra, Portugal, LNCS, Volume 3976, pp. 75-86 (Acceptance ratio = 88/440 = 20%).

[BSL05] (In Chapter 4) - S. Balon, F. Skivée and G. Leduc. Comparing traffic engineering objective functions. *Proc. of ACM CoNEXT student workshop*, pp. 224-225, Toulouse, France, October 24-27, 2005.

[LAB+06] - G. Leduc, H. Abrahamsson, S. Balon, S. Bessler, M. D'Arienzo, O. Delcourt, J. Domingo-Pascual, S. Cerav-Erbas, I. Gojmerac, X. Masip, A. Pescapè, B. Quoitin, S. P. Romano, E. Salvadori, F. Skivée, H. T. Tran, S. Uhlig and H. Umit. An open source traffic engineering toolbox. *Computer Communications*, vol. 29, no. 5, pp. 593-610, March 2006 (ISI IF 2006 = 0.444).

[UQLB06] - S. Uhlig, B. Quoitin, J. Lepropre and S. Balon. Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM Computer Communication Review*, vol. 36, nb. 1, Jan 2006, pp. 83-86 (ISI IF 2006 = 0.578).

[SBD+05] - F. Skivée, S. Balon, O. Delcourt, J. Lepropre and G. Leduc. Architecture d'une boîte à outils d'algorithmes d'ingénierie de trafic et application au réseau GÉANT. In Richard Castanet, editor, *Actes de Colloque Francophone sur l'Ingénierie des Protocoles (CFIP)*, volume Ingénierie des protocoles - Qualité de service, multimédia et mobilité, pages 317-332, Bordeaux, France, 29 Mar.-1 Avr. 2005. Hermès Lavoisier (Acceptance ratio = 33/108 = 30.6%).

2

Overview of Main Intradomain TE Systems

This chapter presents the main ideas developed in intradomain TE techniques found in the literature. Note that the concept of Traffic Engineering and the description of the different involved routing protocols have been introduced in the preceding chapter.

We divide the presented techniques into two main categories: centralized versus distributed. Typically centralized techniques must be run on a server and resulting solutions are then installed on routers. On the contrary, distributed algorithms are directly executed by routers. This implies that distributed techniques can be more reactive, but can be less precise as they are based on local instead of global information. Distributed algorithms are also designed to use less computational resources as these methods have to be run on routers. Traffic engineering techniques can also be classified along other axes: intra-domain versus interdomain, IP versus MPLS or on-line versus off-line.

The extensive presentation of the whole set of Traffic Engineering techniques and algorithms that have been proposed in the literature would be far too long for a chapter in this work. We have intentionally limited our presentation of related works to a selection of the main representative ideas of this research field. This selection is thus obviously subjective and non-exhaustive. We hope we did not forget any important work in the area. We refer to survey papers like [WHPH08, Rex06, MCSA03, YF03] for other analyses and descriptions of the state of the art in Traffic Engineering. In [LM04] the authors specifically analyze Traffic Engineering issues in optical networks based on WDM technology.

2.1 Centralized off-line algorithms

Multicommodity network flow (MCNF) One basic traffic engineering method consists in formulating the routing problem as a Linear Program (LP) ([AMO93]). The routing is optimized for one traffic matrix and one topology. One commodity is assigned to each pair of (ingress, egress) routers. The value of this commodity is the traffic that flows between these two nodes. The algorithm

returns one routing strategy that minimizes the given linear objective function, while respecting the link capacity constraints. The provided objective function can for example be the maximum link utilization.

The routing scheme found by such an algorithm can be implemented in the network with MPLS LSPs. Note that the number of LSPs of the solution can be quite high. This method is often used as benchmark to evaluate the optimality gap of TE heuristic algorithms.

Link Weight Optimizers While Traffic Engineering concepts have been introduced with MPLS, researchers have rapidly developed Traffic Engineering techniques for ASes running only OSPF or IS-IS intradomain protocol. These methods try to find the best possible set of link weights, so that computed shortest paths based on these link weights lead to a good link utilization state. These optimizers have been rapidly introduced in section 1.2.1.1. We will now detail the different variants of this method. OSPF link weight optimization is probably the most studied TE technique.

The first paper dealing with link weight optimizers is [FT00] (extended in [FT04]). The considered problem is combinatorial and they show that optimizing the link weights for a given set of traffic matrices is NP-hard. So the complexity of a link weight optimizer resides in the efficiency of the used search technique. They use a sophisticated heuristic based on tabu search, guiding the search with hashing tables to avoid cycles. They show that on AT&T network with projected traffic demands they obtain a nearly optimal routing scheme, not far from the routing found by an optimal MCNF algorithm. They also test their algorithm on random generated networks and traffic matrices. They conclude that it is possible to obtain a well-engineered network with the OSPF protocol and so MPLS is not really required to perform TE. Similar studies have been performed in [ERP02, BRRT05].

One potential problem with this approach is that the link weights are optimized for one traffic matrix. If the traffic changes, the set of link weights has to be updated. But changing all the link weights can introduce some problems, as the new link weights have to be flooded in the whole AS, then routing tables have to be recomputed and in the meantime packets can be reordered or even be trapped in transient forwarding loops and discarded. In [FT02] Fortz and Thorup study how to limit the number of link weight changes to go from the current routing to a better routing optimized for the current (updated) traffic matrix. They also propose to find a single set of link weights which is optimized for several traffic matrices (for example one representative day TM and one representative evening TM).

In [FT03] Fortz and Thorup analyze link failure scenarios. They adapt their heuristic to find a single set of link metrics which is robust to all single link failures. A similar study is presented in [NSB+03], [Yua03] or [SG05]. In [FRT02] the authors describe the global approach to traffic engineering which includes all the contributions of preceding papers.

Hybrid IP/MPLS optimization In [SBL06] the authors propose a hybrid IP/MPLS approach. Most of the traffic is routed using classical short-

est path routing. For the flows that cannot be routed optimally using shortest path routing, MPLS Label Switched Paths are computed and established. This approach takes advantage of both IP and MPLS technologies and provides a flexible method to traffic engineer a network on a day to day basis. In the practical study, the establishment of a few LSPs is sufficient to obtain a good network state. Other hybrid IP/MPLS approaches are presented in [BALM01, Rie03, MK04b, MK04a, SS04].

Demand Oblivious Routing In [AC03], Applegate and Cohen raise some interesting questions about Traffic Engineering. They study the interactions between the routing and the traffic estimation. They start their work with the assumption that traffic patterns change over time and that it is not generally possible to obtain a good estimate of the current Traffic Matrix. From this point they study the importance of an accurate view of the traffic to obtain good utilizations of the network. They propose an algorithm to obtain a robust routing with limited knowledge of the applicable traffic demands. They test their algorithm on topologies provided by the Rocketfuel Project ([SMW02]). They estimate link capacities with reverse engineering and they generate random traffic matrices for these topologies.

For the tested topologies and traffic matrices, they arrive to surprisingly not so bad *demand oblivious* routing schemes with no knowledge of the traffic matrices. With limited knowledge of the traffic matrix they improve the quality of the solutions found.

COPE An improvement to the techniques of demand-oblivious routing is presented in [WXQ+06]. They state that the problem of oblivious routing is that the network state can be quite far from optimum under *normal* traffic conditions. The authors of [WXQ+06] propose to optimize the routing for measured *(normal)* traffic matrices, while providing a worst case guarantee in case of unpredicted traffic peaks. These TE algorithms are called Common-case Optimization with Penalty Envelope (COPE). They try to take the best from oblivious routing and from prediction based TE.

Two-phase Routing Recently a new two-phase routing scheme has been proposed ([KLOS06]). In the first phase traffic is sent from the source to intermediate nodes, while in the second phase, traffic is sent from the intermediate nodes to the destination. In the first phase the traffic is split on different paths toward intermediate nodes in predetermined proportions. The most important point in two-phase routing scheme is the handling of traffic variability with statistical multiplexing of traffic at intermediate nodes, so that this scheme does not require measurements of traffic in real-time nor reconfiguration of the network in response to traffic changes. This is fully interesting in network with highly variable traffic. In [KLS06] the authors extend this routing scheme to provide resilience, protecting against link failures.

DEFT and PEFT Let us note that routing protocols were not designed for Traffic Engineering. The main weak point of OSPF/ISIS protocols is shortest

path routing combined with equal split on equal cost paths. Some work have proposed to avoid this limitation to perform better traffic engineering for current OSPF/ISIS networks via unequal split on shortest paths ([SGD05]).

There are some recent works arguing that instead of trying to optimize existing routing protocols that were not designed with optimization in mind, researchers should design new routing protocols that are easier to optimize ([HRC07]). In that work the authors also illustrate the design trade-off between network optimizability and the overhead on network resources and complexity.

A recent interesting work in that direction is presented in [XCR07, XCR08]. These works analyze how close to optimal routing a link-state routing protocol can be. In [XCR07] and [XCR08], Xu et al. design an extension to intradomain link state routing protocol where the routers can direct traffic on non-shortest paths, with an exponential penalty on longer paths. These extensions are called Distributed Exponentially-weighted Flow SpliTting (DEFT) and then Penalyzing Exponential Flow SpliTting (PEFT). The proposed modification leads to an easier-to-solve optimization problem.

DEFT keeps the simplicity and robustness of link-state routing protocols as routers compute paths based on link weights but **approaches** the efficiency of protocols that allow arbitrary traffic split on any paths. Contrary to DEFT, PEFT is **proven** to achieve **optimal** TE, and the authors of [XCR08] have demonstrated that link weight computation for PEFT is highly efficient in theory and in practice, while it has not been demonstrated for DEFT.

C-BGP C-BGP[1] is a BGP routing solver ([QPBU05]). The main goal of this simulator is to compute the outcome of the BGP message exchange and BGP decision process run on routers, without simulating unnecessary details. As a result C-BGP is different from the other available open source simulators [Pre, Tya02] which are not able to model networks as large as the Internet, because the simulation becomes quickly untractable as the size of the simulated topology increases, due to the simulated details which have no impact on the final state of the routing tables. At the end of a simulation run, C-BGP provides the interdomain routes selected by BGP routers in a domain.

The route computation of C-BGP relies on an accurate model of the BGP decision process as well as several sources of input data. The model of the decision process takes into account every decision rule present in the genuine BGP decision process as well as the iBGP hierarchy (route-reflectors). The input data required by C-BGP includes intradomain and interdomain information.

C-BGP can be used to test different BGP configurations to perform Interdomain Traffic Engineering.

[1] C-BGP is out-of-scope for this related works section, as we discuss main intra-domain methods and algorithms. Anyway we need to introduce it here as we will use it in some algorithms presented in subsequent chapters.

2.2 On-line constraint-based routing

In this section we review several constraint-based routing (CBR) algorithms. Typically these are TE algorithms that are designed to be deployed in an on-line MPLS-like environment. The path selection is performed at the ingress router, based on a traffic demand request. The request is composed of an ingress node, an egress node and a requested bandwidth. Based on this information the algorithm chooses one path from all the available paths respecting the bandwidth constraints. Then this path needs to be established using for example RSVP.

As the path computation is performed at the ingress router, it needs to obtain some information concerning the available bandwidth in the network. Typically such information can be obtained using TE extensions to traditional link state routing protocols (like [KKY03]).

CSPF, WSP and SWP CSPF (Constraint Shortest Path First) algorithm computes the shortest path (i.e. the path whose sum of link weights is minimal) that satisfies some bandwidth constraints, i.e. all links on the path must have enough free bandwidth to route the demand. It is basically Dijkstra's shortest path algorithm applied on a pruned topology, where links with not enough free bandwidth are removed. This algorithm can be used to route an LSP which needs to reserve a certain amount of bandwidth on the path.

CSPFHopCount is the CSPF algorithm applied with unitary link weights. As a result it computes the path with minimum number of links, or equivalently the minimum number of hops.

WSP (widest-shortest path, introduced in [GOW97]) is an improvement of CSPFHopCount. It tries to balance the traffic in the network. It chooses a feasible path with minimum hop count, and if there are multiple equal minimum hop count paths, it chooses the one with the largest available bandwidth. A dual idea leads to the SWP algorithm (shortest-widest path) where the path with largest residual bandwidth is chosen first, and if multiple widest paths are available, the minimum hop count path is chosen in that set.

DAMOTE DAMOTE [BML03a] (Decentralized Agent for MPLS Online Traffic Engineering) is a routing algorithm used to compute LSPs under constraint. DAMOTE is more sophisticated than a CSPF algorithm. The difference is that DAMOTE finds the path that minimizes a given objective function under bandwidth constraints. Many different objective functions can be used like resource utilization (DAMOTE operates as CSPFHopCount in this case), load balancing (DAMOTE tries to balance the traffic load equally on all links of the network), hybrid load balancing (where long detours are penalized), preemption-aware routing (where induced reroutings are penalized).

DAMOTE is generic for several reasons. Firstly, the score function is a parameter of the algorithm. Secondly, constraints can be combined quite freely. For example, it is possible to define a capacity constraint for different class types (CT) of traffic. In each CT, several preemption levels can be defined. The admission control algorithm will accept a new LSP only if there is enough free bandwidth on all the links of this LSP. The free bandwidth on a link is

computed taking into account the reserved bandwidth of lower preemption level LSPs only (these LSPs are more important). So it is possible to preempt less important LSPs if needed. In this case, DAMOTE is able to choose the "best" set of LSPs to preempt.

DAMOTE computes in an efficient way a near optimal solution. It is also compatible with MAM [LL05] (Maximum Allocation Model) which has been proposed by the IETF MPLS Diff-Serv working group.

MIRA Kar et al. presented an on-line routing algorithm (MIRA) based on the concept of minimum interference ([KKL00]). The amount of interference on a particular source-destination pair (s, d) due to routing a flow between some other source-destination pair is defined as the decrease in the maxflow between s and d. The maxflow value is an upper bound on the total amount of bandwidth that can be routed between two edge nodes ([AMO93]). The minimum interference path for a particular source-destination pair is the path that maximizes the minimum maxflow between all other source-destination pairs. The idea is that a new request must follow a path that does not "interfere excessively" with a route that may be critical to satisfy a future demand. The problem of finding the minimum interference path is proved to be NP-hard. Therefore, Kar et al. proposed to determine appropriate link costs, prune links with insufficient available bandwidth and compute the shortest path in the pruned topology. The definition of link costs involves the notion of critical link for an ingress-egress pair, which is a link belonging in any mincut for that source-destination pair. For each source-destination pair, MIRA computes the maxflow and the set of critical links.

Iliadis and Bauer [IB02] introduced a new class of minimum-interference routing algorithms that reduces the complexity of the path computations. This class of algorithms is called SMIRA (simple minimum-interference routing algorithms).

DORA In [BSI02] Boutaba et al. introduce DORA, a Dynamic On-line Routing Algorithm for computing constrained routes in an MPLS network. The goal of DORA is to efficiently utilize existing network resources and minimize network congestions. The work of DORA is inspired by MIRA algorithm, but performs better in terms of path-setup rejection ratio and rerouting percentage upon link/node failure with much less computation complexity.

DORA algorithm computes paths based on link weights, each link weight being a function of residual bandwidth and also a parameter defined as the path potential value (PPV). The idea is to avoid links that have a low residual bandwidth, but also links that have a high probability to be part of other paths. This latter idea is reflected in the value of the PPV parameter.

2.3 On-line distributed solutions

A minimum Delay Routing Algorithm Using Distributed Computation A visionary work concerning distributed on-line routing was presented in [Gal77] by Gallager. The multipath minimum delay routing problem (of determining at each node the split of traffic that should be routed on each of the router's neighbor outgoing links) is presented as an optimization problem. The author proposed to compute the routing tables in the routers with an iterative algorithm. Nodes apply this algorithm independently of each other. At each iteration the routing tables are updated based on information communicated with neighboring nodes about the marginal delay to each destination. The distributed algorithm is shown to converge to a network state that minimize the overall average cost (e.g. delay) in the network if the traffic is stationary.

MATE In [EJLW01] Elwalid et al. present a multipath adaptive traffic engineering (MATE) mechanism for MPLS networks. The main goal of MATE is to balance the load among multiple paths to avoid network congestion. They propose an algorithm to monitor path quality and detect congestion in order to dynamically adapt the traffic split among multiple paths. MATE does not force intermediate nodes to perform traffic measurements.

TeXCP Another on-line solution to the Traffic Engineering problem is proposed in [KKDC05]. The authors present TeXCP which is an on-line distributed protocol that balances the load on multiple available MPLS Label Switched Paths, that are pre-established in order to provide path diversity. The main idea of TeXCP is similar to the main idea of MATE, but they show with simulations that TeXCP is more effective at balancing the load than MATE and converges faster.

The complete TE solution is composed of several elements. First, multiple paths are computed and established between each pair of (ingress, egress) routers. Then with each of these pairs of routers is associated a TeXCP agent. The TeXCP load balancer at the ingress router splits the traffic towards each egress router on the multiple available paths. The traffic load on each path (i.e. the utilization of the links along each path) is monitored via probes sent from the ingress to the egress and then acked by the egress. This explicit feedback is used to adapt the splitting ratios at regular time intervals, in order to move some traffic from overloaded paths to underloaded paths. The granularity of the traffic split is the flow. Each flow is forwarded on a unique path to avoid reordering of packets which could lead to decreasing the TCP throughput. The time constants and the traffic shifts are carefully designed to avoid traffic oscillations, using control theory.

REPLEX Another interesting work on on-line routing is presented in [FKF06]. The authors propose to use a new traffic engineering protocol. The proposed protocol balances traffic on equal-cost paths computed using the underlying routing architecture, such as OSPF, MPLS or BGP. The traffic split is dynamically adapted and react to traffic changes. The authors use game-theoretical

background to obtain a fast convergence property and avoid oscillations. Information about the traffic conditions along the path can be optionally exchanged between routers. The authors use a distance-vector-like method to obtain a good scalability for the spreading of information. The efficiency of the proposed solution is tested using simulations. The main difference with TeXCP is that REPLEX is not restricted to MPLS-like environments.

3

The TOTEM Toolbox and the Operational Network Dataset

The first part of this chapter (section 3.1) presents the TOTEM toolbox. We have actively contributed to the design and development of this toolbox. Then section 3.2 presents how we have used the toolbox to generate a set of traffic matrices of an operational network. This dataset will be used in subsequent chapters to demonstrate and test the efficiency of new proposed algorithms.

3.1 The TOTEM toolbox

As we have seen, research in the traffic engineering field has been carried out for some years. Solutions exist, but few of these are actually used by operators to manage their network. One reason is that these methods are specifically implemented for research and simulation purposes. It is considered difficult to integrate these methods in an operational environment. One of the main objective of the TOTEM toolbox ([TOT, LAB+06]) is to reconcile the academic and the operational worlds by providing interoperable and user-friendly interfaces with existing tools. This toolbox can also be used by a researcher whose objective is to test, compare and promote his/her own research.

The design of the toolbox also allows different utilization modes. It can be deployed either as an on-line tool in an operational network or as an off-line traffic engineering simulator.

TOTEM is a useful tool for network operators because of the large variety of traffic engineering methods that are integrated. Therefore it is possible to rapidly test and evaluate several engineering solutions.

Note that the description of this chapter is quite limited. A more extensive description of the TOTEM toolbox is available in [LAB+06]. In [BLD+07] the TOTEM toolbox is used to engineer the traffic of an operational network.

3.1.1 Toolbox-related work

Several commercial network optimization toolboxes already exist, e.g., MATE (Cariden) [CAR], Netscope (AT&T) [FGL+00], TSOM (Alcatel) [dBW02], IP/MPLSView (Wandl) [WAN] and SP Guru (Opnet) [OPN]. All these tools are centralized and propose exact and heuristic optimization methods. Most tools are suitable to solve "what-if" scenarios that allow a network operator to evaluate the impact of, e.g., an IGP weight change. All these tools except Netscope also support optimization methods for MPLS networks, including for most of them the computation of backup paths for protection and restoration. Most tools rely on the knowledge of link loads and the existing MPLS LSPs, but MATE also provides a method to derive the traffic matrix from the link loads. The main drawbacks of these commercial tools are their lack of detailed technical public information about their algorithms and the impossibility to upgrade them by new research proposals.

Traffic Engineering Automated Manager (TEAM) [SAC+04] provides an on-line, adaptive approach for automated management of an Internet domain. TEAM is composed of a Traffic Engineering Tool (TET) which adaptively manages the bandwidth and routes in the network, a Measurement and Performance Evaluation Tool (MPET) which measures important parameters in the network, and a Simulation Tool (ST) which may be used by TET to consolidate its decision. TEAM is however only applicable to (DiffServ-based) MPLS networks.

MASCOPT [LSV04] is an open-source network optimization library. It contains a generic graph model, a basic graphical interface, constraint-based routing algorithms taking failures into account, and grooming algorithms for SDH and WDM networks. By contrast to our approach, MASCOPT only provides a library, not a complete toolbox. In that sense, it is comparable to the generic tools and topology manager present in TOTEM.

TOTEM is different from all other network optimization tools. To the best of our knowledge, TOTEM is the only open-source toolbox for intradomain and interdomain traffic engineering of IP and MPLS networks, providing stable and robust methods for IGP metric optimization, primary and backup LSP routing, and BGP simulations. These methods can be easily compared, combined and extended.

3.1.2 Software architecture

The toolbox contains different modules:

- **Topology module:** contains the set of classes related to the network topology which allows for example to add or remove some links, to add or remove some LSPs, to check some properties on the network (e.g. the connectivity), or to obtain some statistics (e.g. the network utilization);
- **Traffic matrix module:** contains some functionalities related to traffic matrices like reading files, checking the consistency of the traffic matrix with respect to the link capacities and generation of traffic matrices;
- **Scenario module:** contains the classes related to simulation scenarios

providing the ability to read, execute or generate scenarios (explained below);

- **Algorithm repository:** This is the central part of the toolbox. It contains all the traffic engineering algorithms[1]: IGP-WO (an IGP Link Weight Optimizer), C-BGP, DAMOTE, MIRA, different versions of CSPF, ...
- **Chart module:** contains some functionalities related to chart generation. This module allows the user to automatically generate charts using various data sources;
- **Graphical User Interface (GUI):** provides an easy interface to test the toolbox methods. This interface displays a view of a network topology and allows a user to see the effect of an action taken on the network on the link loads, e.g. a link failure, a change of an IGP metric or a change of the LSP routing policy.

3.1.3 Simulation scenarios

To simplify the use of the toolbox in simulation mode, we set up a kind of scripting language by means of XML scenario files. The content of a XML scenario file is a sequence of events that will be executed by the toolbox. We defined a set of basic events (”linkDown”, ”linkUp”, ”LSPCreation”, ”loadDomain”, etc.) which allow to build complex scenarios. An example of a scenario file could be:

- load a topology and a traffic matrix;
- display the resulting link loads using a SPF algorithm;
- optimize the IGP weights using the Link Weights Optimizer;
- display the link loads with updated weights.

The language defined by the XML scenario files can be easily extended, i.e. it is easy to write new events. These new events can be based on already integrated algorithms or on new algorithms that are plugged in the toolbox during runtime.

3.1.4 Data flows in the toolbox

The process to engineer a network from data collection to analysis report is described in Figure 3.1. The first step is to collect data and aggregate them to produce a topology, one or more traffic matrices and a BGP routing table. The second step is to create a simulation scenario (of section 3.1.3) that will control the toolbox execution. The toolbox will simulate the scenario and produce some reports (text files or simple graphs). With this process, it is simple to simulate link failures, traffic matrix evolution or IGP metric optimization and to analyze the impact on link loads. It is also possible to replace the simulation scenario with the use of the Graphical User Interface.

[1] These were presented in details in chapter 2

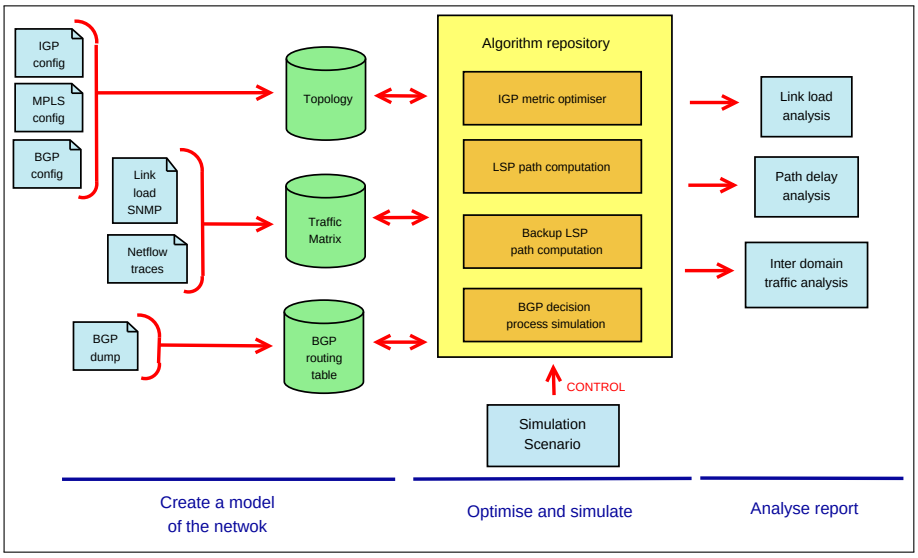

Figure 3.1: Traffic engineering analysis using the toolbox

3.2 Obtaining a dataset from a real network

In chapters 4 to 9 we present several new Traffic Engineering methods. When such new methods were developed it was necessary to evaluate their computational efficiency and the quality of the solutions found. For that purpose it would have been possible to generate random topologies and random traffic matrices to simulate how the new methods would behave if they were applied in these randomly generated networks. The problem with such simulations is that one can always say that such simulations do not prove anything. Because random networks and random traffic matrices are not real networks and real traffic matrices. This is the main reason why we have decided to try to obtain real data from an operational network. Typically traffic matrices can be computed based on traffic measurements [FGL+01, MTS+02, ZRDG03] or may reflect explicit reservations like Service Level Agreements (SLAs) negotiated with network users.

Thanks to a collaboration with UCL (Université Catholique de Louvain), we have obtained the required measurements. This traffic information has been measured on a multi-gigabit operational network that spreads over the European continent and is composed of about 25 nodes and 40 bidirectional intradomain links. Link capacities range from 155Mbps to 10Gbps. It is a transit network that has two providers connected with about 10 interdomain links, has other peer ASes connected with about 15 shared-cost links, and has more than 25 customer ASes, which are mainly single-homed. The total traffic exchanged is about 10 Gbps on average.

We had access to about one month of traces (one month of year 2005), one BGP dump per day and one sampled netflow file for each ingress router. With these data we have generated 2,512 aggregated traffic matrices (each matrix is an average over 15 minutes). Some of these induce a low load on the network

while some induce a high load. This whole set of traffic matrices is representative of the traffic on the studied network.

3.2.1 Building the Interdomain Traffic Matrix

We have obtained the NetFlow data collected on each router of the network. Basically, netflow data contains information about flows passing through the network. NetFlow data is dumped every 15 minutes. Flows are cut by NetFlow timer (120 sec). Every flow is recorded by the ingress node, i.e. the node by which the flow enters the network.

We have chosen to aggregate NetFlow data by source prefix and destination prefix. As the route of a flow in an IP network is determined by longest prefix match, we do not lose any useful information (for our usage of course). Indeed we aggregate flows using BGP information, i.e. the advertised BGP prefixes (we explain how to get them in the next paragraph). When aggregated, NetFlow data results in simple text files (one per node) containing for each pair of source and destination prefixes a corresponding flow size in bytes (with a sample rate of 1/1000 in our case). To build the intradomain traffic matrix from this aggregated information, we need the ingress node and the egress node for each source / destination prefix pair. The ingress node is simply the node on which the flow has been recorded. To compute the egress node (which is the BGP next-hop), it is more complicated and we need the C-BGP simulator which has been integrated in the toolbox. At this stage, the aggregated netflow information grouped by ingress node is called the interdomain traffic matrix.

3.2.2 Collecting BGP data

BGP has been presented in chapter 1. The iBGP configuration used in the network we consider is an iBGP full-mesh. To collect BGP traces, a monitoring machine has been installed inside the network. This monitoring machine is part of the iBGP full-mesh and records all the exchanged BGP messages into BGP traces. The BGP traces we have are daily dumps containing all the routes received by the monitoring machine.

3.2.3 From the interdomain traffic matrix to the intradomain traffic matrix

Now that we know how BGP traces are recorded and how the data have been aggregated, we need to know how to compute the egress node for each destination prefix. To this end, we need to know the (interdomain) routing table of each router. We do not have this information in the BGP dump. We will use the C-BGP routing solver from the TOTEM toolbox to recompute the routing tables for each router based on the BGP dumps. C-BGP is able to replay all message exchanges and all decision processes that took place in the iBGP full-mesh, so that each node will have a best route to each destination.

To replay all the exchanges of BGP messages, we have first to enhance the topology of the network with iBGP and eBGP session information. We added

an iBGP full-mesh. To add eBGP sessions, we have used the BGP dump. When a router has sent to the monitoring machine its best route telling that it has a route towards an external prefix received through a given external peer, we know that this router has an eBGP session with this peer. We checked that, using this technique, we had all the eBGP sessions present on the network.

As there are about 150000 prefixes, which is huge to replay in C-BGP, we grouped them into clusters, i.e. we group prefixes that are announced in exactly the same way (i.e., on the same node, from the same peer, with the same BGP parameters[2]), and we only advertise one of the prefixes belonging to one cluster. This allows us to advertise only about 400 prefixes into C-BGP. For each prefix for which we need to know the next-hop on a given node, we find the corresponding advertised prefix belonging to the same cluster and retrieve the routing table of the concerned node where we find the next-hop. This next-hop is the egress point of the network for this destination prefix.

3.2.4 Properties of the dataset

Now we will analyze the properties of the dataset.

3.2.4.1 Total traffic

First we analyze the total traffic on the network (the sum of all the traffic crossing the network). Figure 3.2 presents the total traffic crossing the network for the whole dataset. We can see that there is a diurnal pattern (five weekdays followed by two weekend days). On that figure the traffic seems to be quite "regular", except for some "down" spikes.

Figure 3.3 presents the total traffic on week 2. On the whole dataset, there are five significant "down" spikes (TM IDs: 1257, 1395, 1443, 1866 and 2127). These spikes are probably due to some problems in the data collection process.

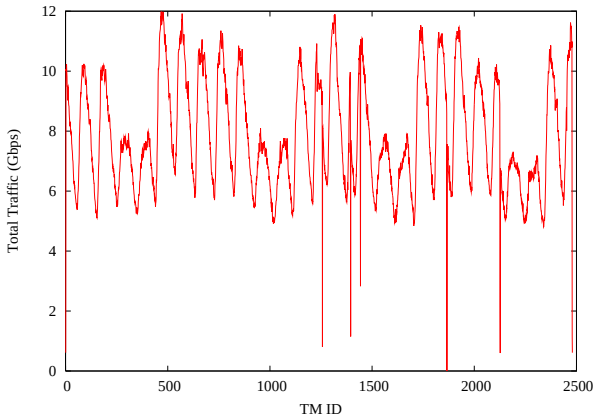

Figure 3.2: Total traffic on the network: The whole dataset

[2]Here, by BGP parameters, we mean the local preference, the AS path, the MED, the origin (IGP/EGP/INCOMPLETE) and the next-hop address.

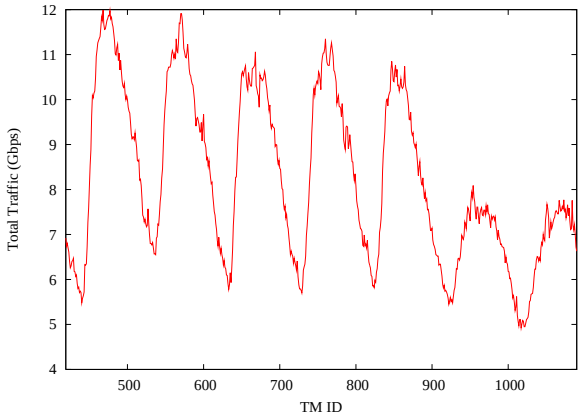

Figure 3.3: Total traffic on the network: Week 2

3.2.4.2 Source traffic fanouts

Now we analyze the source traffic fanouts. We define the source traffic fanout for node N_1 as the total traffic injected in the network by node N_1 divided by the total traffic crossing the network. Similarly the destination traffic fanout for node N_1 is the total traffic whose exit point is N_1 divided by the total traffic crossing the network. If both the source and destination traffic fanouts are constant for all the nodes over time, it is quite good for traffic engineering. Indeed this means that if we engineer the routing scheme for one (past) traffic matrix and that the traffic increases at a later stage, it will probably increase equally on all ingresses and egresses, if the traffic fanouts remain constant. If the traffic fanouts are permanently changing it is more difficult to predict the future traffic matrices.

We do not present all the source traffic fanouts plots as this would require too much space. We will present some typical plots but also plots that present some interesting patterns. Some source fanouts are quite constant (e.g. figure 3.4) while some are more changing (e.g. figure 3.5). We can observe some spikes on the curves, but also some abrupt changes in the mean fanout values (e.g. figure 3.6). These abrupt changes let us suppose a change of ingress point for some traffic flows. This may be due to BGP routing changes (for example a new BGP route announcement or a change in BGP configuration like MED or local pref), to a link weight change in a neighboring domain leading to a hot-potato rerouting forcing an egress point change in that neighboring domain, or to any interdomain traffic engineering technique used in this neighboring domain. Note that we have no information to determine which of these possible events was the cause of each particular ingress change.

An interesting non-regularity in the source traffic fanout is the increase from about 6% to about 11% during the period TM $182 \rightarrow 555$ for the node N_4 (figure 3.7). This transient change is different from others because this is not linked to a significant decrease of traffic fanout for any other node.

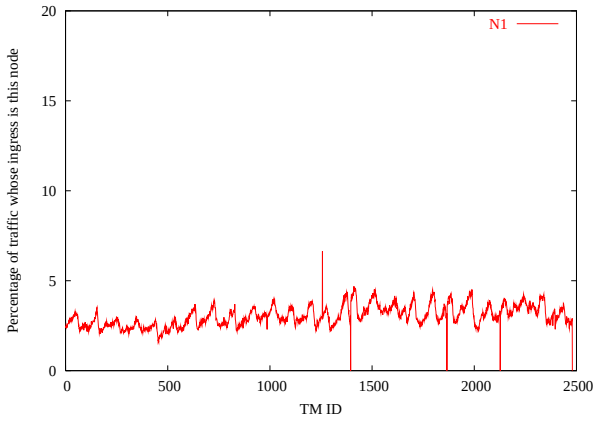

Figure 3.4: Source fanout for N_1

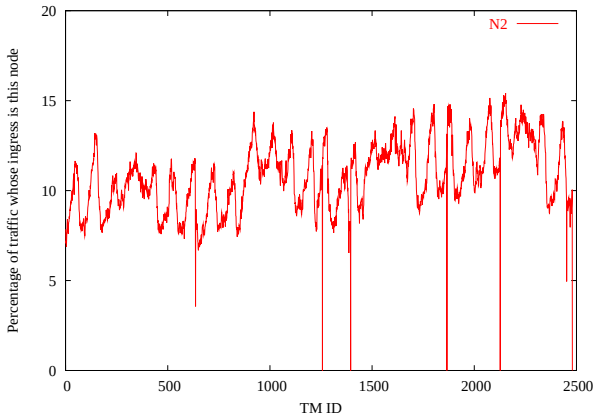

Figure 3.5: Source fanout for N_2

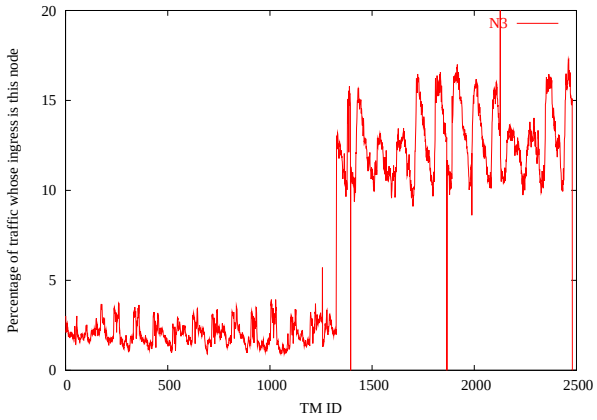

Figure 3.6: Source fanout for N_3

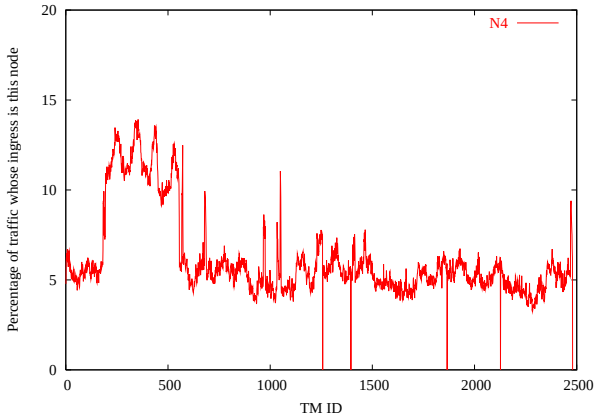

Figure 3.7: Source fanout for N_4

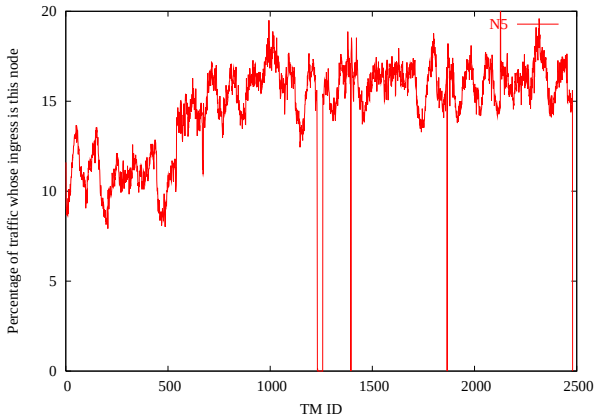

Figure 3.8: Source fanout for N_5

3.2.4.3 Destination traffic fanouts

Now we analyze the destination traffic fanouts. They are roughly similar to the source traffic fanouts. We have observed abrupt change in mean destination traffic fanouts on figures 3.10 and 3.11.

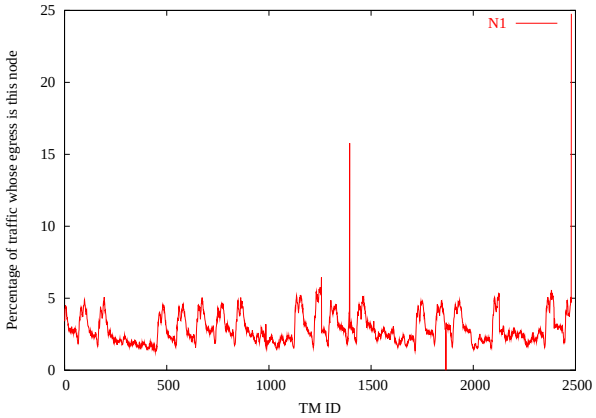

Figure 3.9: Destination fanout for N_1

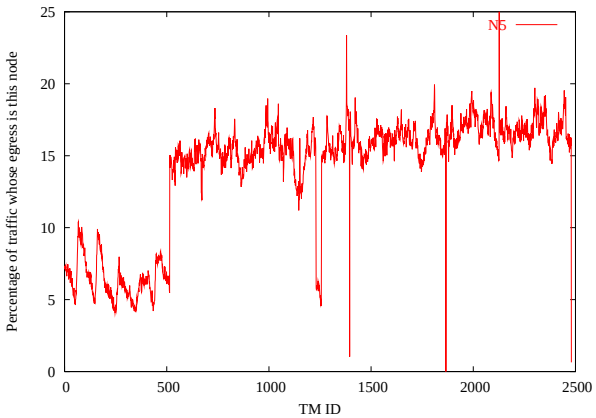

Figure 3.10: Destination fanout for N_5

The transient increase in the traffic fanout for N_6 (figure 3.11) between TM 183 to 555 is not linked to a significant decrease in the fanout of another node. We can remember that we had a similar increase in the source traffic fanout for N_4. This means that a huge amount of data has been transferred from N_4 to N_6 during that time interval.

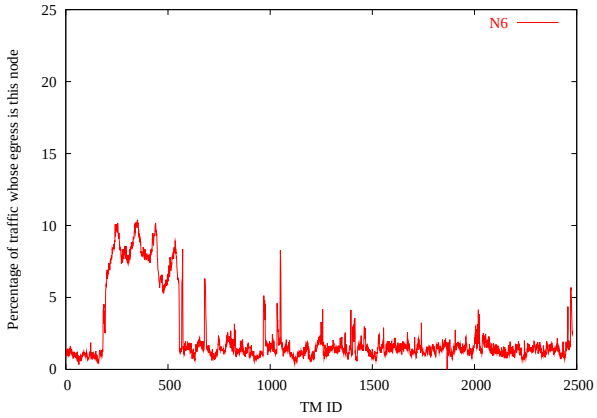

Figure 3.11: Destination fanout for N_6

3.3 Conclusion

In the first part of this chapter we have presented the TOTEM toolbox. Developing the TOTEM toolbox was the main objective of the TOTEM project which has been funded by the Direction Générale des Technologies, de la Recherche et de l'Energie (DGTRE) of the Walloon government. Three research teams were involved in the TOTEM project (the Research Unit in Networking of ULg and two teams of UCL headed by O. Bonaventure and B. Fortz). At ULg we have developed the core of the toolbox and integrated intra-domain IP and MPLS methods including the algorithms presented in the subsequent chapters of this thesis. The two teams at UCL have integrated inter-domain tools and intra-domain IP methods. I have actively contributed to the design and implementation of the toolbox, together with other colleagues at ULg including Olivier Delcourt, Jean Lepropre, Gael Monfort and Fabian Skivée.

We have also cooperated with other European research teams (Delft University of Technology, FTW Austria and University of Naples "Federico II") in the framework of the E-NEXT Network of Excellence (funded by the European Union). I have particularly worked with the University of Naples "Federico II" on the integration of their implementation of MIRA and also on the integration of the toolbox in the management tools of their testbed.

In the second part of this chapter we have presented the dataset of the operational network which will be used in subsequent chapters to test the quality of new proposed methods and algorithms. We have also analysed the properties and anomalies of the source and destination traffic fanouts, which will be later used in chapter 6, to analyse why some simulations lead to bad TE performance.

4

Mathematical Formulation of TE Goals and Objectives

In this chapter we compare and evaluate how well-known and novel network-wide objective functions for Traffic Engineering (TE) algorithms fulfill TE requirements. To compare the objective functions we model the TE problem as a linear program and solve it to optimality, thus finding for each objective function the best possible target of any heuristic TE algorithm. We show that all the objective functions are not equivalent and some are far better than others. Considering the preferences a network operator may have, we show which objective functions are adequate or not.

4.1 Introduction

Here we model the traffic engineering routing problem as follows. Given the topology of the network to be engineered and an estimate of the traffic matrix to be routed on it, the problem (see figure 4.1) is to find a routing scheme that optimizes the network, with the joint goal of good user performance and efficient use of network resources. The way classical algorithms fulfill this objective is not clear. Indeed, many algorithms try to optimize their home-made objective functions which are said (but not proven) to reflect traffic engineering objectives. The foundations of all these objective functions are related, but could lead to quite different results, as we see in our simulations.

Some in-depth reflection and comparison studies of traffic engineering objective functions are needed. In many research papers, the quality of a new traffic engineering algorithm is evaluated regarding one specific objective function. If the algorithm obtains a good score, it is considered as good. But this is only meaningful if the objective function really reflects the traffic engineering goals. Furthermore, when analyzing published papers it is difficult to figure out if the merits of a given TE method is due to its objective function or its heuristic algorithm. To fill this gap, we provide an independent comparison of many

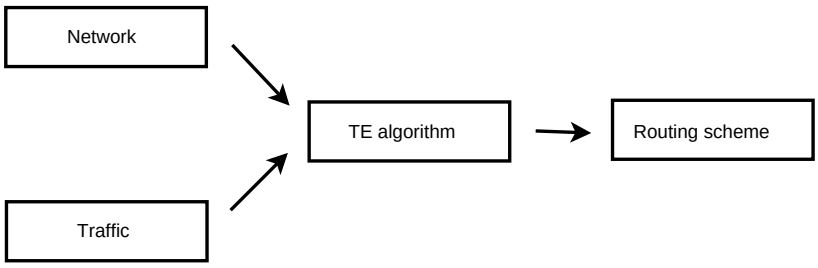

Figure 4.1: TE problem

objective functions found in the literature[1].

To compare all the different objective functions, we will minimize (or maximize) each of these functions on the same topology and traffic matrix and analyze if the routing scheme we obtain really reflects general Traffic Engineering goals. One important point is that we have used some real topologies and real traffic matrices to run our tests, which is not the case of many research papers.

Section 4.2 introduces some Traffic Engineering metrics that we will use in our objective function comparison. Section 4.3 presents existing TE algorithms and related objective functions. We discuss the foundation of these objective functions and why they were introduced. In section 4.4, we construct LP (Linear Programming) models of these objective functions. These models are used to compare all the presented functions on different networks. Then we analyze the results of simulations, highlighting the merits and/or shortcomings of each objective function. Finally, section 4.5 concludes the chapter.

4.2 Traffic Engineering objectives

A network is modeled as a directed graph, $G = (N, A)$ whose nodes and arcs represent routers and links. Each arc has a capacity c_a. Traffic on the network is represented by a traffic matrix D that with every pair (s, t) of nodes associates the value of the traffic demand, i.e. the traffic that flows from node s to node t.

Basically, the graph G and the traffic matrix D are the inputs of the problem. The TE algorithm has to find *good* paths between each pair of source and destination nodes to route corresponding traffic flow. The definition of *good* paths is related to what we want to optimize on the network. Generally, a good set of paths will be one that optimizes a pre-defined objective function.

Once the paths are chosen, we can associate with each arc a load l_a, which is the total load on the arc, i.e. the sum over all demands of the amount of traffic sent over a. The utilization of a link a is $u_a = l_a/c_a$. The available bandwidth on link a is $ABW_a = c_a - l_a$.

Finally, we define θ_{st} as the maximum flow that can be sent from node s to

[1]Note that in [Vas79] Vastola presents an early investigation of the impact of cost functions in routing optimization. The author compares two different measures of delay in the network.

node t in the residual network, i.e. when the whole traffic matrix is routed on the network.

4.2.1 View of the IETF

Before presenting different objective functions of the literature, we will present the requirements for Traffic Engineering as introduced by the IETF. In [AMA+99], we can read that *Traffic Engineering (TE) is concerned with performance optimization of operational networks.* Awduche et al. divide Traffic Engineering performance objectives in two categories: traffic oriented or resource oriented.

Traffic oriented performance objectives include the aspects that enhance the QoS of traffic streams (...) including: minimization of packet loss, minimization of delay, maximization of throughput, and enforcement of service level agreements, (...) peak to peak delay variation, loss ratio, and maximum packet transfer delay. Concerning resource oriented performance objectives, *it is generally desirable to ensure that subsets of network resources do not become over utilized and congested while other subsets along alternate feasible paths remain underutilized. (...) a central function of Traffic Engineering is to efficiently manage bandwidth resources.*

We can also read that *minimizing congestion is a primary traffic and resource oriented performance objective.* The type of congestion that can be addressed through Traffic Engineering is *when traffic streams are inefficiently mapped onto available resources; causing subsets of network resources to become over-utilized while others remain underutilized.* This type of congestion can be reduced by adopting load balancing policies. *The objective of such strategies is (...) to minimize maximum resource utilization. (...) When congestion is minimized through efficient resource allocation, packet loss decreases, transit delay decrease, and aggregate throughput increases.*

We can clearly deduce from this that one objective that should be respected by all the objective functions is to minimize maximum resource utilization. In addition to this, some other objectives can be taken into account. These additional objectives should be related to congestion minimization.

4.2.2 Discussion on TE objectives

Typically, on-line algorithms have different objectives than off-line ones. On-line schemes usually try to minimize the probability of blocking future requests, while off-line ones try to minimize the load or the utilization of the links, or try to maximize available bandwidth. To some extent, minimizing the link utilization (which is a relative measure) tends to maximize the available bandwidth (which is an absolute measure) on the links, thus also reducing the blocking probability of future requests. Clearly, these objectives are closely related, but no solid basis exists to choose one among all.

We will consider TE metrics at three different levels, which are a link, an OD pair[2] and the network. We will present and justify the foundation of the TE metrics at each level. We will differentiate metrics whose goal is to improve

[2]OD stands for Origin Destination.

the quality of the network given the present traffic (e.g. minimize the delay) from metrics whose goal is to maximize the acceptance of future traffic on the residual network (e.g. maximize residual max-flow).

At the *link level*, we should minimize delay and utilization. We should also maximize the available bandwidth on this link (which corresponds to the notion of residual max-flow for a link). The delay of a link is composed of three components: the propagation delay ($delay_p$) which is a constant value, the transmission delay (inversely proportional to the link capacity) and the queuing delay which increases with the link load. If we take the delay to be the average delay of an M/M/1 queue, the mean queuing + transmission delay ($delay_{q+t}$) of link a is given by $Delay_a = \frac{mean\_packet\_size}{c_a - l_a}$. For a M/M/1 queue, all the percentiles/quantiles are also proportional to this value. On high capacity links, this delay is significant only if the link load is approaching the link capacity. Figure 4.2 summarizes the relations between link parameters.

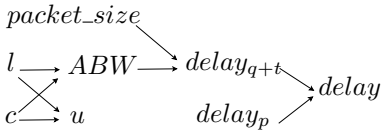

Figure 4.2: Link parameters

At the level of an *OD pair* of nodes, we should minimize the path delay, i.e. the sum of the delays of all the links on the path. Minimizing this delay can increase the quality of service perceived by the users of the network. We should also minimize the maximal link utilization on the corresponding path. Indeed, the maximal link utilization has a particular meaning. For example a maximal link utilization (u_{max}) of 50% means that we can double all the traffic before having a link fully loaded (if we keep the same routing scheme), while a value of 20% means that we can multiply all the traffic by 5. In fact this value ($\frac{1}{u_{max}}$) is a lower bound because a change in the routing scheme may allow increasing this value. Finally, the residual max-flow between an OD pair of nodes should be maximized. Indeed, this value represents the maximal size of a future request that can be routed on the network between these nodes.

We have now some ideas of TE metrics to be optimized for a link or for an OD pair. But to be really useful in TE algorithms we have to generalize these concepts to the *whole network*. There are many ways to proceed. For example, considering link utilizations, one can minimize the maximal link utilization (u_{max}) as for the OD level[3]. But the minimization of the maximal link utilization works poorly in some cases. Indeed if there is a real bottleneck in the network, i.e. a link whose utilization cannot be decreased by changing the routing scheme, it is important to minimize also the utilization of other links. One way to proceed can be to minimize the mean link utilization. Considering the delay to be minimized on the whole network, we can compute the mean link delay (each link being weighted by its load or not) or the mean path delay (each path being weighted by its corresponding traffic). The unweighted mean path

[3]We can prove that the routing scheme that achieves the minimal value of maximum link utilization also provides the optimal value concerning the factor by which it is possible to multiply the current traffic matrix. In this case, this factor can be computed as $\frac{1}{u_{max}}$.

	Metric characterizing good current state	Metric characterizing likely good future
$\text{Link}_{(a)}$	$Delay_a$	u_a, ABW_a
$\text{Path}_{(s,t)}$	$\sum_{a \in \mathcal{P}(s,t)} Delay_a$	θ_{st}, $max_{a \in \mathcal{P}(s,t)} u_a$
Network	$\frac{\sum_{a \in A} Delay_a}{\lvert A \rvert}$, $\frac{\sum_{a \in A} l_a \times Delay_a}{AllTr}$	$min_{(s,t)} \theta_{st}$, $max_{a \in A} u_a$ $\sum_{(s,t)} \theta_{st}$

Table 4.1: TE metrics summary

delay seems less relevant to us. The following demonstration shows that the weighted mean path delay is equivalent to the weighted mean link delay. This demonstration highlights that it is possible to compute the mean path delay without path information. This is an important result from a computational point of view. Indeed it is less complex to compute a sum over all links than a sum over all paths of all possible OD pairs of nodes.

$$
\begin{aligned}
&MeanDelay = \frac{1}{\sum_{(s,t)} D(s,t)} \sum_{(s,t)} D(s,t) \sum_{a \in \mathcal{P}(s,t)} delay_a \\
&= \frac{1}{AllTr} \sum_{(s,t)} D(s,t) \sum_{a \in \mathcal{P}(s,t)} delay_a \\
&= \frac{1}{AllTr} \sum_{(s,t)} D(s,t) \sum_{a \in A} \delta_{a \in \mathcal{P}(s,t)} delay_a \\
&= \frac{1}{AllTr} \sum_{a \in A} delay_a (\sum_{(s,t)} D(s,t) \delta_{a \in \mathcal{P}(s,t)}) \\
&= \frac{1}{AllTr} \sum_{a \in A} l_a \times delay_a
\end{aligned}
$$

$\mathcal{P}(s,t)$ denotes the path from s to t[4], $\delta_{a \in \mathcal{P}(s,t)}$ is equal to one if link a belongs to $\mathcal{P}(s,t)$ and 0 otherwise. $AllTr$ denotes the sum of all traffics of the network ($\sum_{(s,t)} D(s,t)$). It is a constant for a given problem.

Considering max-flows (θ), it is possible to maximize the minimal residual max-flow. But as for the maximal link utilization, the minimal max-flow can be blocked by a set of bottleneck links. So we should also maximize the sum of all max flows (instead of the min value). We could also associate with each max-flow a weight related to its corresponding traffic demand.

As it could be interesting to maximize the sum of residual max-flows, it could be interesting to maximize the sum of the available bandwidths over all the links of the network. However, we can notice that it is equivalent to minimizing the sum of the loads over all the links of the network. Indeed, $max \sum_{a \in A} ABW_a = max \sum_{a \in A} (c_a - l_a) = max(\sum_{a \in A} c_a - \sum_{a \in A} l_a) \equiv min \sum_{a \in A} l_a$, as the sum of all the capacities of the network is invariant.

Table 4.1 presents a summary of TE metrics introduced in this section.

4.2.3 How to measure the quality of a solution?

Table 4.2 presents the TE metrics we will use to evaluate the quality of the routing solutions in the simulation section. As presented in the preceding subsection, it is clear that the maximum link utilization (u_{max}) is a good TE metric.

[4]Here, we assume that there is only one path used from s to t, but the demonstration can be easily generalized if there are multiple paths.

u_{max}	$max_{a \in A} u_a$
u_{mean}	$\frac{1}{\lvert A \rvert} \sum_{a \in A} u_a$
u_{per90}	$Nb\{a \in A \lvert u_a > u_{per90}\} = \lvert A \rvert .90\%$
ABW_{min}	$min_{a \in A}(c_a - l_a)$
l_{mean}	$\frac{1}{\lvert A \rvert} \sum_{a \in A} l_a$
$delay_{mean}$	$\frac{1}{AllTraffic} \sum_{a \in A} l_a \times \frac{packet\_size}{c_a - l_a}$
θ_{min}	$Min_{(s,t)} \theta_{st}$
θ_{tot}	$\sum_{(s,t)} \theta_{st}$

Table 4.2: Definition of TE metrics

In addition, the mean link utilization (u_{mean}), the 90^{th} percentile (u_{per90}), the minimal available bandwidth (ABW_{min}) and the mean load (l_{mean}) will be used. u_{per90} is defined so that 90% of the links have a utilization under u_{per90}. We think that the weighted mean queuing + transmission delay of the network ($delay_{mean} = \frac{1}{AllTraffic} \sum_{a \in A} l_a \times \frac{packet\_size}{c_a - l_a}$) is also an important variable. We will also consider the minimum max flow ($\theta_{min} = Min_{(s,t)} \theta_{st}$) and total max flow ($\theta_{tot} = \sum_{(s,t)} \theta_{st}$) of the residual topology. The total max flow gives an idea of the throughput, i.e. which amount of traffic can be accepted on the residual network. This is not exactly the amount of bandwidth that can be routed on the residual network because all max-flows are computed independently of each other and thus all the flows are not in competition for the residual bandwidth. But this can still give a good idea of the residual throughput[5].

4.3 Presentation of different objective functions

4.3.1 Fortz

In [FT00], B. Fortz et al. introduce the concept of link weight optimizers, which has been presented in chapter 2. A cost is associated with each link of the network. This cost (ϕ_a) is a function of the link load (see figure 4.3) where for all $a \in A$, $\phi_a(0) = 0$ and

$$\phi'_a(x) = \begin{cases} 1 & \text{for} & 0 & \leq & x/c_a & < & 1/3 \\ 3 & \text{for} & 1/3 & \leq & x/c_a & < & 2/3 \\ 10 & \text{for} & 2/3 & \leq & x/c_a & < & 9/10 \\ 70 & \text{for} & 9/10 & \leq & x/c_a & < & 1 \\ 500 & \text{for} & 1 & \leq & x/c_a & < & 11/10 \\ 5000 & \text{for} & 11/10 & \leq & x/c_a & < & \infty \end{cases}$$

The objective function they try to minimize is the sum over all links of this cost ($\phi = \sum_{a \in A} \phi_a$). We will later refer to this objective function as *Fortz*.

[5]The amount of traffic that can be routed on the residual network is in fact the sum over all links of the available bandwidth. Indeed one obvious (and degenerated) solution to the max throughput problem is to associate traffic only with the pairs of nodes that are located at the extremities of a link. We can associate with these pairs the available bandwidth on the corresponding link.

The idea behind ϕ_a is that it is cheap to send flow over an arc with a small utilization. As the utilization approaches 100%, it becomes more expensive, for example because we get more sensitive to bursts. (...) when the utilization goes above 100% the penalty gets so high that this should never happen. This function clearly tries to minimize maximum resource utilization. They prefer to minimize the sum over all links instead of the maximum over all links because *even if there is a bottleneck link that is forced to be heavily loaded, this objective function still cares about minimizing the loads in the rest of the network.* We have noticed that this function, though empirical, could be seen as a linear approximation of $\frac{l_a}{1-u_a}$ (also drawn on figure 4.3). At low link utilization, $1-u_a \approx 1$ and $Fortz \approx min \sum_{a \in A} l_a$, while at high utilizations, $\frac{1}{1-u_a}$ becomes significant, leading to a load balancing policy (avoiding links with high utilization). To summarize $Fortz$ can be seen a a function trying to minimize the max link utilization, the size of the paths and the mean queuing + transmission link delay, thus mixing metrics characterizing good current state with metrics characterizing likely good future. There is no OD pair consideration in this objective function.

Many papers (as [WYXN05] for example) have reused this objective function.

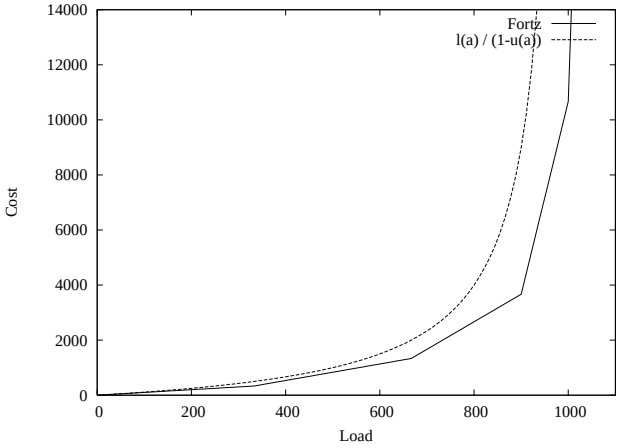

Figure 4.3: ϕ_a = cost of a link a for which $c_a = 1000$.

4.3.2 MIRA

In [KL00], Kodialam et al. introduce the concept of minimum interference routing. They propose an objective function which is a weighted sum of the maxflows over all possible source-destination pairs on the residual topology. Their online algorithm is a heuristic that tries to maximize this objective function. Formally, the objective function to maximize is $\sum_{(s,t)} \alpha_{st}\theta_{st}$, where α_{st} is a weight associated with the ingress-egress pair (s,t) (recall that θ_{st} represents the maximum flow that can be sent from node s to node t in the residual network). The weights associated with ingress-egress pairs are administrative weights that determine the relative importance of the ingress-egress pairs to the network administrator. If these are not given, we can take all ingress-egress pairs to have the same weight. Behind this objective function, the goal is to minimize the

blocking probability of a future new request, without information about it. The idea is that if the maxflow between one source and one destination decreases, this means that the maximum request that can be accepted between these two nodes decreases as well.

To summarize, the $MIRA$ objective function is characterizing likely good future. There is no embedded metric characterizing good current state. We will see later in the simulations the implications of this fact.

In [SWBW03], some improvements are proposed to the on-line heuristic algorithm of MIRA. But the high level goal is the same as MIRA, i.e. to minimize the blocking probability. We thus consider that this algorithm tries to optimize the same kind of objective function. They evaluate their algorithm by way of simulation scenarios, comparing their algorithm to other ones in terms of blocking probabilities.

4.3.3 Blanchy

In [BML03b], Blanchy et al. present an online traffic engineering algorithm to optimize a load balancing objective function. The algorithm is a heuristic that gives very good solutions in a short time. The pure load balancing objective function is $\sum_{a \in A}(u(a) - u_{mean})^2$ with $u_{mean} = \frac{1}{|A|}\sum_{a \in A} u(a)$, the mean link utilization in the network. *This function is the variance on the link utilization (the relative link load) and, as such, represents the deviation from the optimal load balancing situation. (...) The main problem with this load balancing function is that the only thing it tries to do is to flatten the relative load throughout the network. It will not matter if some of the paths go a long way around in order to achieve a better load-balancing. (...) We must then try to limit the length of the paths by adding a kind of "shortest path length" term to the objective function.* To limit the length of the paths of a pure load balancing function, they add a "shortest path" term and arrive at the following objective function: $\sum_{a \in A}(u(a) - u_{mean})^2 + \alpha \sum_{a \in A}(u(a))^2$. It is interesting because *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. The weighted factor α allows to give more importance to one aspect or to the other.*

In [BML03b], we can also read that *Load balancing the network should ideally produce a network with a homogeneous blocking probability by source-destination pair. Hopefully, it could also lower the overall blocking probability in comparison with a classical shortest path (in terms of number of hops). On the other hand, using such traffic engineering techniques sometimes favours detours and might logically increase further the load inside the network (again compared with a minimum hop routing). Clearly, there must be a compromise between load-balancing and traffic minimization.*

This objective function does not directly include TE metrics we introduced in section 4.2.2. It does not include a delay contribution and there is no consideration about OD pairs. The traffic minimization term tries to minimize the size of the paths. To some extent, as the utilization is squared, this function also tries to minimize the maximal link utilization, because the non-linearity gives more penalty to loaded links.

4.3.4 Delay

In [EJLW01], Elwalid et al. associate a cost with each link. They try to minimize the total cost which is the sum over all links of the link cost. The cost of a link is a function of the link load. They assume that this function is convex. They say that a natural choice for the link cost is the delay so that their network-wide cost function is defined as $MeanDelay = \sum_{a \in A} \frac{1}{c_a - l_a}$. In section 4.2.2 we called this function the (unweighted) mean link delay, if we do not take the propagation delay into account. We introduce a new delay objective function (referred to as $WMeanDelay$) which is $\sum_{a \in A} \frac{l_a}{c_a - l_a}$, the weighted mean delay. Note that this objective function can also be formulated using only u_a as $WMeanDelay = \sum_{a \in A} \frac{u_a}{1 - u_a}$. These objective functions are metrics characterizing good current state. On figure 4.4, we can see the delay objective function for one link and also its piecewise linear approximation.

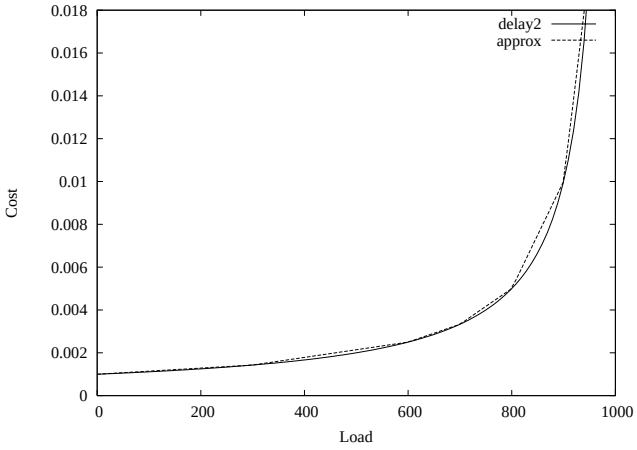

Figure 4.4: Delay of a link a for which $c_a = 1000$.

4.3.5 Degrande

In [DHdLVPdB03], Degrande et al. propose to maximize an objective function which is composed of four terms: $C_F.F + C_T.T + C_B.B - C_U.U$ with F(airness), T(hroughput), B(alance) and (network) U(tilisation) the four performance parameters to be optimized and C_F, C_T, C_B and C_U the objective coefficients. They say that these coefficients *can be defined such that* $C = \frac{C'}{X_0}$ *with* X_0 *the value of the corresponding objective for shortest path routing and* C' *a value that prioritizes the different objectives (*10^9, 10^6, *...). The choice of them depends on the type of traffic in the network and optimization strategy.* Fairness and Throughput are traffic oriented objectives while Balance and Utilization are resource oriented objectives. Balance is defined as: $B = 1 - u_{max}$. Network utilization is defined as $U = \sum_{a \in A} u_a$. We will not consider Fairness and Throughput in our formulation because it is not possible to express these in our LP formulation. The balance is a metric characterizing likely good future. The utilization term will minimize the size of the paths. There is no OD pair consideration and no delay contribution in this objective function.

	Score Function (to be minimized)
Fortz	$\sum_{a \in A} \phi_a$
MIRA	$-\sum_{(s,t)} \theta_{st}$
Blanchy	$\sum_{a \in A}(u_a - u_{mean})^2 + \alpha \sum_{a \in A}(u_a)^2$
MeanDelay	$\sum_{a \in A} \frac{1}{c_a - l_a}$
WMeanDelay	$\sum_{a \in A} \frac{l_a}{c_a - l_a}$
InvCap	$\sum_{a \in A} u_a$
u_{max}	u_{max}
Degrande	$C_B.u_{max} + C_U.\sum_{a \in A} u_a$
MinHop	$\sum_{a \in A} l_a$

Table 4.3: Summary of objective functions

Some papers (like [MU03] for example) only try to minimize the maximum link utilization. This is equivalent to *Degrande* objective function where $C_B = 1$ and $C_U = 0$. We will refer to this objective function as u_{max}.

Degrande objective function where $C_B = 0$ and $C_U = 1$ is a function which minimize $U = \sum_{a \in A} u_a$. This objective is also minimized by a classical SPF routing considering link weights equal to the inverse of their capacities. In fact, inverse capacity routing (recommended by CISCO) gives the optimal value of U. We will thus refer to this objective function as $InvCap$. We prove this in chapter 5.

4.3.6 Summary

Clearly, all the presented objective functions are related, while quite different. A first difference is that some of them use only absolute values of the load l (like $MIRA$), some only relative values u (like *Blanchy*, $WMeanDelay$, or *Degrande*) and finally some use both (like $Fortz$, or $MeanDelay$).

Table 4.3 presents a summary of all the presented objective functions. *MIRA*'s function is used with $\alpha_{st} = 1, \forall (s,t)$. For *Blanchy*, we have to fix the α parameter. For *Degrande*, we have to fix C'_B and C'_U. In the table, we have added the cost function called *MinHop*. This function simply minimizes the total load over all the links of the networks ($\sum_{a \in A} l_a$). This function is minimized by a SPF routing considering a weight of 1 for each link (what we call a min hop routing). See chapter 5 for details.

We can point out that at low load, $1 - u \approx 1$ and $c - l \approx c$ and thus $Fortz \approx MinHop$ while $WMeanDelay \approx InvCap$.

4.4 Simulations

In order to compare all the objective functions, we will model the traffic engineering routing problem as a linear program (LP) and solve it to optimality for all the presented objective functions. In this formulation, all the flows can be arbitrarily split. Obviously, this cannot be really implemented in a network, but

can be approached with MPLS routing and to some extent with ECMP. This assumption allows us to formulate the problem as a linear program (which is easy to solve to optimality) instead of a mixed integer program (which cannot be solved to optimality in a reasonable time). The LP formulation will be used to solve the routing problem to optimality and compare the solutions obtained for every objective function. We have used an LP node-link formulation :

Variables :

$x_a^{st} \geq 0$
$load_a$
$utilization_a$

Constraints :

$$\sum_{a \in In(n)} x_a^{st} - \sum_{a \in Out(n)} x_a^{st} = \begin{cases} D^{st} & \text{if } n = t \\ -D^{st} & \text{if } n = s \\ 0 & \text{otherwise} \end{cases} \qquad s,t,n \in N$$

$$load_a = \sum_{(s,t)} x_a^{st} \qquad a \in A$$

$$utilization_a = \frac{load_a}{capacity_a} \qquad a \in A$$

$$load_a \leq capacity_a \qquad a \in A$$

where x_a^{st} is the part of traffic from node s to node t that flows on arc a, $load_a$ is the load of arc a, $In(n)$ is the set of incoming arcs of node n, $Out(n)$ is the set of outgoing arcs of node n. The first constraint expresses that for each commodity (s, t), the outgoing traffic of a node is equal to the incoming traffic of this node, except if the node is the source or the destination of this commodity. In this case, it is equal to the traffic associated with this commodity. The second constraint expresses that the load of a link is equal to the sum of all the traffic using this link. The third constraint expresses that the utilization of a link is its load divided by its capacity. The last constraint expresses that the load cannot exceed the capacity of the link.

We will not write the formulation of all the objective functions, because this would take too much space. Instead, we explain clearly how they can be reproduced. *Fortz* linear formulation is expressed in [FT00]. For $MIRA$, we use a classical *max flow* formulation for each pair of nodes. For *Blanchy*, the square function is approximated by a piecewise linear approximation in the range $[-1, 1]$, as shown in figure 4.5 (for 8 linear pieces). $MeanDelay$ and $WMeanDelay$ are approximated by convex piecewise linear functions. *Degrande* and $MinHop$ are linear so they can be expressed easily, without modification. We will not present $MeanDelay$ in our result tables because we have noticed similar results than for $WMeanDelay$ (noted *Delay* in the tables). For *Degrande* function, we use $C'_B = 10^3$ and $C'_U = 1$ (as in [DHdLVPdB03]) and for *Blanchy*, $\alpha = 3$ (which seems to provide good results).

4.4.1 Simulation description

We made our simulations on three different networks. The first topology was generated in the TOTEM toolbox [LAB+06] using Waxman's method [Wax88].

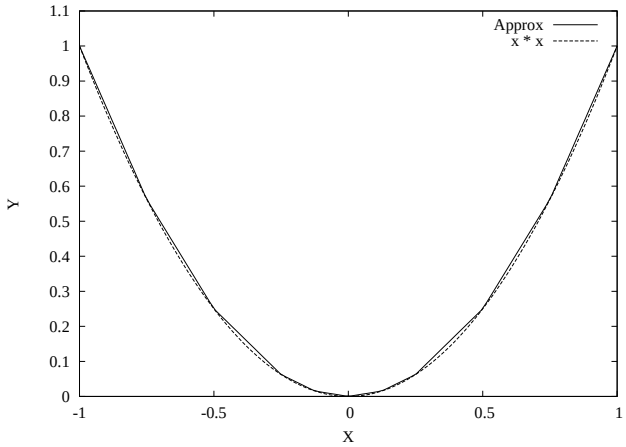

Figure 4.5: Piecewise Linear approximation of the square function between -1 and 1 that we use in *Blanchy* objective function.

This topology is composed of 25 nodes and 50 full-duplex links. We set the value for parameters α and β to 0.15 and 0.2. We have generated a random traffic matrix for this topology. The second topology is the operational network whose dataset was presented in chapter 3. From the whole dataset we have selected one representative TM to run the simulations of this chapter. The last topology is the US research network (Abilene). This American research network is illustrated on figure 4.6. It is composed of 11 nodes and 14 bidirectional links of 10 Gbps each. We have used netflow data measured on the network to build a realistic traffic matrix. We have run our simulation on two traffic matrices per topology: the actual one (TM) and the double of it (2TM) (where each OD component has been multiplied by 2, trying to obtain an estimate of a potential increased future traffic matrix).

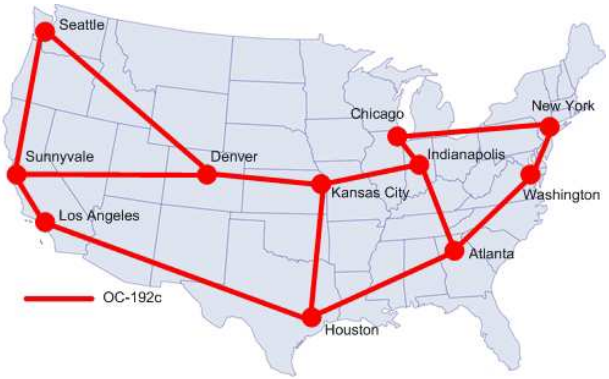

Figure 4.6: Abilene network

4.4.2 Results

We have to keep in mind that we made some linear approximation of some objective functions. The original function could give slightly different results in some cases. Also, some (non-linear) objective functions give different results depending on the load of the links. So, the particular traffic matrices and networks on which we made the tests can have its influence as well. Notice that $InvCap$ and $MinHop$ objective functions do not provide exactly the same routing scheme as classical shortest path first algorithm with inverse capacity or unitary metrics. Indeed, our LP model of these objective functions allows extensive and non-equal flow splitting (which is not the case in classical OSPF or ISIS implementations). So our results may present better solutions than the ones obtained by shortest path first algorithms. We have also noticed a negative point for some objective functions: multiple routing schemes achieve the optimal objective function value (especially for u_{max} objective function). By default, the LP solver returns one of these solutions, at random. As we did not want random values in our tables, we have added a small delay contribution to these objective functions ($MIRA$, $InvCap$, u_{max}, $Degrande$ and $MinHop$) so that the LP solver returns a "good" solution from the set of possible equivalent routing schemes. So we should keep in mind that these objective functions could lead to worse results than the ones presented in this section if we do not add this delay contribution and if we do not allow arbitrary flow splitting.

Tables 4.4, 4.5 and 4.6 give the values of the TE metrics at the optimum for each objective function on Waxman, the operational and Abilene networks. We have removed the θ_{min} metric from the tables because all the objective functions obtained the optimal value. We have also removed the $delay_{mean}$ metric because corresponding values were either very small or infinite (when $ABW_{min} = 0$). Indeed, we do not take the propagation delay into account and the link capacities are huge. This implies that all the delay values are almost equivalent, because negligible when compared to the propagation delays. But although all the delay values (except infinite values, of course) are tiny, we can point out that the $Delay$ objective function gives good results for all the TE metrics on all the topologies. This is because the delay objective embeds most TE concerns (load, utilization, available bandwidth) and even though the queuing delays are most often negligible, they become non-linearly sufficiently high when the load approaches the capacity to enforce load balancing. u_{per90} is not present in table 4.6 because there are few links in Abilene network and thus this metric is not very significant.

Objective function	u_{max} %		u_{per90} %		u_{mean} %		ABW_{min} Mbps		l_{mean} Mbps		θ_{tot} Mbps	
	TM	2TM	TM	2TM	TM	2TM	TM	2TM	TM	2TM	TM	2TM
Fortz	(1.14)	(1.14)	(1.28)	(1.33)	(1.26)	(1.21)	(0.67)	*(0.55)*	**(1.03)**	**(1.05)**	**(0.97)**	**(0.95)**
MIRA	*100*	*100*	(1.40)	(1.48)	(1.17)	(1.16)	*0.0*	*0.0*	(1.15)	(1.10)	**6504**	**5012**
Blanchy	(1.22)	(1.23)	**26.0**	**50.0**	(1.13)	(1.12)	(0.88)	**531**	(1.12)	(1.11)	**(0.96)**	**(0.94)**
Delay	(1.20)	**(1.08)**	(1.17)	(1.20)	**(1.04)**	(1.11)	**882.0**	**(0.95)**	(1.16)	(1.11)	**(0.97)**	**(0.95)**
InvCap	*(2.07)*	*100*	(1.55)	(1.61)	**15.7**	**31.5**	**882.0**	*0.0*	(1.21)	(1.20)	**(0.98)**	**(0.96)**
u_{max}	**34.9**	**69.7**	(1.15)	(1.20)	**(1.07)**	(1.12)	(0.74)	*(0.57)*	(1.17)	(1.11)	**(0.97)**	**(0.95)**
Degrande	**34.9**	**69.7**	(1.35)	(1.39)	**(1.05)**	**(1.05)**	(0.74)	*(0.57)*	(1.19)	(1.18)	**(0.97)**	**(0.95)**
MinHop	*100*	*100*	(1.29)	(1.43)	(1.27)	(1.25)	*0.0*	*0.0*	**781**	**1578**	**(0.97)**	**(0.95)**

Table 4.4: Results on network of 25 nodes (Waxman topology). The table contains absolute optimal values (in bold, green, without parentheses), or relative non-optimal values (between parentheses) with respect to the optimal one. The values that are less than 10% from the optimal value are bold. Finally the values that are 2 times worse than the optimal one are in italic and red. For each metric, we present the values for the actual traffic matrix (TM) and for the doubled traffic matrix (2TM).

Objective function	u_{max} %		u_{per90} %		u_{mean} %		ABW_{min} Mbps		l_{mean} Mbps		θ_{tot} Gbps	
	TM	2TM	TM	2TM	TM	2TM	TM	2TM	TM	2TM	TM	2TM
Fortz	(1.18)	(1.13)	(1.63)	(1.17)	(1.17)	(1.14)	(0.89)	(0.56)	**(1.00)**	**(1.04)**	**(0.99)**	**(0.98)**
MIRA	(1.41)	*100*	(1.63)	(1.65)	**(1.07)**	**(1.09)**	(0.75)	*0.0*	**(1.03)**	**(1.05)**	**4331**	**4027**
Blanchy	(1.16)	(1.15)	**14.2**	**28.5**	**(1.07)**	(1.11)	**(0.90)**	*(0.50)*	(1.24)	(1.23)	**(0.99)**	**(0.97)**
Delay	**(1.04)**	**(1.02)**	(1.32)	(1.21)	**(1.01)**	**(1.02)**	**(0.97)**	**(0.92)**	(1.15)	(1.17)	**(0.99)**	**(0.99)**
InvCap	(1.18)	**(1.09)**	(1.51)	(1.49)	**6.9**	**13.8**	(0.89)	(0.69)	(1.19)	(1.19)	**(0.99)**	**(0.98)**
u_{max}	**38.4**	**76.9**	(1.56)	(1.21)	**6.9**	**(1.01)**	**95.7**	**36.0**	(1.20)	(1.15)	**(0.99)**	**(0.99)**
Degrande	**38.4**	**76.9**	(1.51)	(1.49)	**6.9**	**13.8**	**95.7**	**36.0**	(1.19)	(1.19)	**(0.99)**	**(0.98)**
MinHop	(1.36)	*100*	(1.76)	(1.60)	(1.16)	(1.20)	(0.78)	*0.0*	**262**	**525**	**(0.99)**	**(0.98)**

Table 4.5: Results on the operational network. See the legend of table 4.4 to understand these values.

Objective function	u_{max} %		u_{mean} %		ABW_{min} Gbps		l_{mean} Gbps		θ_{tot} Tbps	
	TM	2TM	TM	2TM	TM	2TM	TM	2TM	TM	2TM
Fortz	**(1.07)**	**(1.07)**	(1.11)	**(1.08)**	**(0.96)**	(0.83)	(1.11)	**(1.08)**	**(0.98)**	**(0.96)**
MIRA	(1.75)	*100*	**10.2**	**21.2**	(0.60)	*0.00*	**1.02**	**2.12**	**2.05**	**1.58**
Blanchy	(1.34)	(1.41)	**(1.07)**	**(1.06)**	(0.82)	*0.03*	**(1.07)**	**(1.06)**	**(0.99)**	**(0.98)**
Delay	(1.43)	**(1.07)**	**(1.04)**	**(1.08)**	(0.77)	(0.84)	**(1.04)**	**(1.08)**	**(0.99)**	**(0.96)**
InvCap	(1.73)	*100*	**10.2**	**21.1**	(0.60)	*0.00*	**1.02**	**2.11**	**2.05**	**(0.99)**
u_{max}	**35.0**	**70.1**	(1.13)	(1.11)	**6.50**	**2.99**	(1.13)	(1.11)	**(0.98)**	**(0.95)**
Degrande	**35.0**	**70.1**	(1.13)	**(1.09)**	**6.50**	**2.99**	(1.13)	**(1.09)**	**(0.98)**	**(0.96)**
MinHop	(1.73)	*100*	**10.2**	**21.1**	(0.60)	*0.00*	**1.02**	**2.11**	**2.05**	**(0.99)**

Table 4.6: Results on the Abilene network. See the legend of table 4.4 to understand these values.

We start our analysis with table 4.4, which presents results for the topology generated using Waxman's model. We can see that all the objective functions are not equivalent. $MinHop$ is given for comparison purposes (it gives the lowest achievable value for l_{mean}) but is clearly not a good objective function on its own. Indeed, it leads to a high value of u_{max} which is a very important concern. The lowest achievable value for u_{max} is given by the u_{max} function which only optimizes this variable. The lowest achievable value of the u_{mean} variable is given by $InvCap$. This function is not very good on its own because it leads in this case to a high u_{max} value. The combined $Degrande$ is a very good objective function on this topology. Indeed, it gives nearly optimal values for all the metrics. $Blanchy$, $Fortz$ and $Delay$ are quite good. We notice also that $MIRA$ is good except for u_{max} which is 100% (and thus $ABW_{min} = 0$ and $delay_{mean}$ is infinite). We analyze this fact as follows. $MIRA$ is based on max-flows (and only on max-flows). Suppose that we have two routes in the network for a particular OD pair of nodes. The value of the residual max-flow will be the same if we route all the traffic on one route or if we route half of it on each route. This is the cause of the bad load balancing policy and the high value of u_{max} given by $MIRA$.

To better discriminate the $Degrande$, $Delay$ and $Blanchy$ functions we can analyze the results corresponding to the doubled traffic matrix. In this case, $Blanchy$ has a quite high value of u_{max}, while $InvCap$ leads to a fully loaded link ($u_{max} = 100\%$). $Degrande$ and $Delay$ are in this case the best objective functions. $Fortz$ is also quite good in this situation.

On table 4.5 we can see the results for the operational topology. $Blanchy$ obtains good values for all the metrics and the best value of u_{per90}. $MIRA$ logically gives the optimum for the θ_{tot} variable, which is its objective function. We remark that many other objective functions give values close to this optimal θ_{tot} value. On the operational network, we consider that the best compromise is $Degrande$ because it gives almost optimal values for all the variables except u_{per90}. Both $Delay$ and $Blanchy$ are quite good and give better results for u_{per90}. $Fortz$ improves l_{mean} at the expense of all the other variables. $MIRA$ and $MinHop$ give high values regarding u_{max}.

We have noticed on the Abilene network that there is less variation between the values of our metrics. But we have still pointed out the performance of $Delay$ and $Degrande$ which are the best objective functions of these simulations.

One last important point is the fact that at low load, we can see that $Fortz$ is approaching the optimal value of l_{mean}, the objective of $MinHop$, while $Delay$ is approaching the optimal value of u_{mean}, the objective of $InvCap$. This confirms the approximation we made in section 4.3.6.

To conclude this section, we analyze table 4.7 which presents the good ($\surd$) and bad ($\bullet$) metrics for each objective function at low and high load[6]. On this table, we see that $Degrande$, $Delay$ and $Fortz$ are the best because these have no red point.

[6]In this table, $\surd\surd$ is used to denote the optimal value and $\pm$ to denote a value which is not bad, but which is not as good as $\surd$ values.

Objective Function	u_{max}		u_{per90}		u_{mean}		ABW_{min}		l_{mean}		θ_{tot}	
	LL	HL	LL	HL	LL	HL	LL	HL	LL	HL	LL	HL
Fortz	√	√	√	√	√	√	±	±	√	√	√	√
MIRA	•	•	•	•	√	√	•	•	√	√	√√	√√
Blanchy	√	•	√	√	√	√	√	•	√	√	√	√
Delay	√	√	√	√	√	√	√	√	√	√	√	√
InvCap	•	•	√	±	√√	√√	±	•	√	√	√	√
Degrande	√√	√√	√	√	√	√	√	√	√	√	√	√

Table 4.7: Metrics At Low Load (LL) and High Load (HL)

4.5 Conclusion

In this chapter, we have shown how well-known network-wide objective functions reflect requirements for Traffic Engineering. As our results reflect, they are not equivalent. We have shown the power of some functions and the weaknesses of others. We have outlined that, although the transmission + queuing delay is often negligible, choosing this delay as objective function gives good results for almost all TE metrics. It is not that surprising considering that almost all TE link metrics feed into the delay (see figure 4.2).

The best objective functions are *Delay* and *Degrande* on the tested topologies. We have a preference for *Delay* because it does not need any configuration or parameter. *Fortz* is quite good also in all the situations, while having performance somewhat under *Delay* and *Degrande*. *Blanchy* has good results also, except for highly loaded networks. *MIRA* gives good solutions concerning the total residual max flow, but this function gives bad results concerning the maximal link utilization.

This study provides an objective basis to select an objective function when designing a new Traffic Engineering routing algorithm. It may also be useful to revisit existing TE algorithms to make them work with the objective functions that best match the various TE concerns we have studied.

5

Approaching Optimal Intradomain TE?

In this chapter we propose a new method to approach optimal Intradomain Traffic Engineering routing. The method consists in dividing the traffic matrix into N sub-matrices, called strata, and route each of them independently. We propose two different implementations of our method in routers. Our method can also be used to compute a very precise approximation of the optimal value of a given objective function for comparison to heuristic Traffic Engineering algorithms. For this application, our algorithm is very efficient on large topologies compared to an LP formulation.

5.1 Introduction

We consider the traffic engineering routing problem as defined in preceding chapter.

We propose to divide the traffic matrix into N equal sub-matrices, called strata, for which the routing scheme can be independently chosen. The sum of the N strata is obviously equal to the original traffic matrix. In section 5.2, we present the objective functions we consider in this chapter, while our method can be used with other objective functions. Section 5.3 presents the first derivative of the objective functions we consider. In section 5.4, we propose an algorithm to compute one routing scheme per strata using the derivatives computed in section 5.3. Section 5.5 presents the execution of our new algorithm on a simple example. Section 5.6 shows that the total resulting routing scheme is close to the optimum for large N. We have also noticed that in practice, a low value of N is sufficient to obtain a routing scheme very close to the optimal one. In section 5.7 we evaluate the efficiency of our algorithm compared to LP formulation. Finally, in section 5.8 we propose two methods to implement our routing scheme in routers. The first one is to establish N MPLS full-meshes in the network. The second one is to use the IGP Multi-Topology functionality ([PSS08, PMR+07]). Our algorithm provides the paths of all the LSPs for the MPLS solution and the N sets of metrics for the Multi-Topology Routing

solution.

5.2 Objective functions

5.2.1 Presented objective functions

We consider the objective functions presented in table 5.1. We reuse the notation of chapter 4. These functions must be minimized by TE algorithms. All these objective functions can be written under the form of $\sum_{a\in A} f_a(l_a)$ and $f_a(x)$ are convex. $MinHop$ is the objective function minimized by a minimum hop routing. A minimum hop route is a route with minimal number of links. $InvCap$ objective function is minimized by a shortest path routing considering a metric of $\frac{1}{c_a}$ for each link a[1]. $WMeanDelay$ minimizes the weighted mean delay and is a good TE objective function, as we have seen in chapter 4. $MeanDelay$ is the (unweighted) mean delay. Finally, $NonLinearFortz$ is a non linear function whose linear approximation is introduced in [FT00] by Fortz and Thorup[2].

	Objective Function
MinHop	$\sum_{a\in A} l_a$
InvCap	$\sum_{a\in A} u_a$
WMeanDelay	$\sum_{a\in A} \frac{l_a}{c_a - l_a} = \sum_{a\in A} \frac{u_a}{1-u_a}$
MeanDelay	$\sum_{a\in A} \frac{1}{c_a - l_a}$
NonLinearFortz	$\sum_{a\in A} \frac{l_a}{1-u_a}$

Table 5.1: Objective functions

5.3 The first derivative of objective functions

The goal of the TE algorithm is to find good paths between each pair of nodes. The idea of our algorithm is that a good path minimizes the increase of the score function due to the routing of some traffic on this path. The increment of the cost function of using one particular link on the path of an OD pair is $\sum_{a\in A} f_a(l'_a) - \sum_{a\in A} f_a(l_a)$, if l_a and l'_a are the loads of link a before and after routing some traffic on it. Let x be the load of the link a^* we consider and dx the increment of traffic on this link. The increment is 0 for all other links. Thus $\sum_{a\in A} f_a(l'_a) - \sum_{a\in A} f_a(l_a) = f_{a^*}(x+dx) - f_{a^*}(x)$. The increment is thus $\frac{f_{a^*}(x+dx) - f_{a^*}(x)}{dx}$ per unit of traffic flowing on this link a^*. If we suppose that the traffic increment on this link is sufficiently small, we can assume that the increment of the cost function is $\lim_{dx\to 0} \frac{f_{a^*}(x+dx) - f_{a^*}(x)}{dx} = \frac{\partial f_{a^*}(x)}{\partial x}$. Hereunder we compute this first derivative for all the presented objective functions.

[1] This is proven in section 5.4.

[2] We have already presented this objective function in chapter 4

$$
\begin{aligned}
MinHop \quad & \frac{\partial f_a(x)}{\partial x} = \frac{\partial x}{\partial x} = 1 \\
InvCap \quad & \frac{\partial f_a(x)}{\partial x} = \frac{1}{c_a} \\
WMeanDelay \quad & \frac{\partial f_a(x)}{\partial x} = \frac{c_a}{(c_a - x)^2} \\
MeanDelay \quad & \frac{\partial f_a(x)}{\partial x} = \frac{1}{(c_a - x)^2} \\
NonLinearFortz \quad & \frac{\partial f_a(x)}{\partial x} = \frac{c_a^2}{(c_a - x)^2}
\end{aligned}
$$

We notice that for $MinHop$ and $InvCap$ objective functions, the first derivative does not depend on x, while it is the case for all the other functions.

5.4 Algorithm

We claim that for $MinHop$ and $InvCap$ objective functions, if we use Dijkstra's Shortest Path First (SPF) algorithm taking as link metric the first derivative of the objective function then we obtain the optimal routing scheme, i.e. we find for this routing the minimal value of the objective function.

Theorem 5.1. *A SPF algorithm where link metrics are equal to $\frac{1}{c_a}$ (resp. 1) leads to the minimal value of the InvCap objective $= \sum_{a \in A} u_a$ (resp. MinHop objective $= \sum_{a \in A} l_a$) independently of the traffic matrix.*

Proof. We want to minimize the $InvCap$ objective function: $\sum_{a \in A} u_a$. We have

$$
\begin{aligned}
\sum_{a \in A} u_a &= \sum_{a \in A} \frac{\sum_{(s,t)} \delta_{a \in \mathcal{P}_{st}} D_{(s,t)}}{c_a} \\
&= \sum_{(s,t)} D_{(s,t)} \sum_{a \in \mathcal{P}_{st}} \frac{1}{c_a}
\end{aligned}
$$

if $\mathcal{P}_{st}$ is the path from node s to node t and $\delta_{a \in \mathcal{P}_{st}} = 1$ if link a is on the path from s to t and 0 otherwise[3]. $\sum_{(s,t)} D_{(s,t)}$ is constant and so to minimize $\sum_{a \in A} u_a$, we have to minimize $\sum_{a \in \mathcal{P}_{st}} \frac{1}{c_a} \forall s, t$ which is minimized by SPF algorithm taking $\frac{1}{c_a}$ as link metrics. □

For the $InvCap$ objective function, we thus find the CISCO recommendation for IGP metric setting ($\frac{1}{c_a}$). The proof can be easily adapted for the $MinHop$ function. In this case $\frac{1}{c_a}$ is replaced by 1, which justifies its name. We see that the $MinHop$ routing thus minimizes the total load of the network, as expressed by the $MinHop$ objective function.

Theorem 5.1 cannot be applied directly with the last three objective functions of table 5.1 because they are non linear with respect to the link loads.

[3]If we consider ECMP, $\delta_{a \in \mathcal{P}_{st}}$ is the fraction of traffic sent on a.

5.4.1 Dividing the traffic matrix

To approximate infinitesimal increments of traffic for the last three objective functions, we propose to divide the traffic matrix into N equal traffic sub-matrices called strata. The algorithm first computes the paths of the first stratum. Then the algorithm takes into account the link loads induced by the routing of this first stratum to compute the paths of the second stratum, and so on, until the N^{th} stratum. Let OBJ_n be the value of the objective function at step n and OBJ_N its value at the end of the process.

Lemma 5.1. *For large N,*

$$OBJ_n - OBJ_{n-1} \approx \sum_{(s,t)} \frac{D_{(s,t)}}{N} \sum_{a \in \mathcal{P}^n_{(s,t)}} f'_a(l_a^{n-1})$$

Proof.

$$\begin{aligned} OBJ_n &= OBJ_{n-1} + \sum_{a \in A} \left[f_a(l_a^n) - f_a(l_a^{n-1}) \right] \\ &= OBJ_{n-1} \\ &\quad + \sum_{a \in A} \left[f_a(l_a^{n-1} + \frac{\sum_{(s,t)} D_{(s,t)} \delta_{a \in \mathcal{P}^n_{(s,t)}}}{N}) - f_a(l_a^{n-1}) \right] \end{aligned}$$

As

$$lim_{\epsilon \to 0} f_a(l_a^{n-1} + \epsilon) = f_a(l_a^{n-1}) + \epsilon \times f'_a(l_a^{n-1})$$

we have for large N:

$$OBJ_n \approx OBJ_{n-1} + \sum_{a \in A} \sum_{(s,t)} \frac{D_{(s,t)}}{N} \delta_{a \in \mathcal{P}^n_{(s,t)}} f'_a(l_a^{n-1})$$

□

Theorem 5.2. *For large N, the routing paths of the n^{th} stratum that minimize $OBJ_n - OBJ_{n-1}$ are the shortest paths according to the following link metrics: $w_a^n = f'_a(l_a^{n-1})$.*

Proof. This result is derived directly from Lemma 5.1 with $\mathcal{P}^n_{(s,t)}$ being the shortest paths with w_a^n as link metrics. □

From this theorem we can conclude that if the traffic matrix is split into a large number of strata, each new stratum can be routed so as to minimize the increase of the objective function, simply by routing the n^{th} stratum along the shortest paths with link metrics w_a^n. The succession of n such steps, all minimizing the increase of OBJ, is thus expected not to depart too much from the optimum.

The algorithm we propose to implement this method is thus the following (see Algorithm 5.1). First compute the paths using SPF algorithm with metrics

Algorithm 5.1: Divide TM into N sub-matrices

$l_a^0 \leftarrow 0 \quad \forall a \in A;$
$n \leftarrow 1;$
while $n \leq N$ **do**
  $w_a^n \leftarrow \left.\frac{\partial f_a(x)}{\partial x}\right|_{x=l_a^{n-1}} \quad \forall a \in A;$
  $l_a^n \leftarrow l_a^{n-1} + \sum_{(s,t)} \delta_{a \in SPF^n_{(s,t)}} \frac{D_{(s,t)}}{N} \quad \forall a \in A;$
  $n \leftarrow n + 1;$
end
$l_a \leftarrow l_a^N \quad \forall a \in A;$

equal to $w_a^1 = \left.\frac{\partial f_a(x)}{\partial x}\right|_{x=0}$. Route the first stratum using these paths and recompute the new metrics considering the load introduced by this routing. Compute the paths for the second stratum using these updated metrics, and so on, until the N^{th} partial traffic matrix. $\delta_{a \in SPF^n_{(s,t)}}$ is equal to 1 if a belongs to the shortest path from s to t considering the set of metrics w_a^n and 0 otherwise[4].

5.5 Simple example

In this section, we present an example of our algorithm running on a simple topology and its limit when $N \to \infty$. We highlight why it is not optimal and when this non-optimality occurs. In this example we use the $MeanDelay$ objective function but the other functions would lead to the same kind of results.

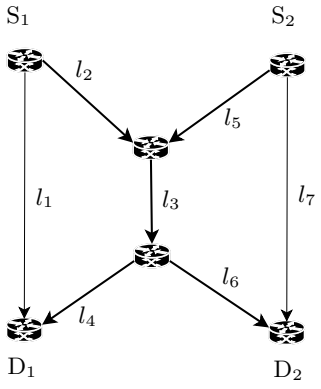

Figure 5.1: Simple Example Topology

Consider the topology of figure 5.1. $c_{l_1} = 11$ Mbps, $c_{l_3} = 10$ Mbps and $c_{l_7} = 9$ Mbps. $c_{l_2} = \infty$, $c_{l_4} = \infty$, $c_{l_5} = \infty$, $c_{l_6} = \infty$[5]. $D_{(S_1,D_1)} = 10$ Mbps and $D_{(S_2,D_2)} = 1$ Mbps. The traffic matrix is empty for all other (source,

[4]If we consider ECMP, $\delta_{a \in SPF^n_{(s,t)}}$ is the fraction of traffic sent on a considering w_a^n as link metrics.

[5]Instead of infinite capacities, we can also use capacities $\gg \{c_{l_1}, c_{l_3}, c_{l_7}\}$

destination) pairs of nodes. There are two possible paths for traffic from node S_1 to node D_1, the left-hand path l_1 and the right-hand path $l_2 l_3 l_4$. For the traffic from node S_2 to node D_2, the two possible paths are the left-hand path $l_5 l_3 l_6$ and the right-hand path l_7. We can easily compute that the optimal routing scheme occurs if S_1 sends 5.5 Mbps of traffic on its left-hand path and 4.5 Mbps of traffic on its right-hand path while S_2 sends all its traffic on its right-hand path. In this case the value of the objective function is equal to $\frac{1}{c_{l_1}-l_{l_1}} + \frac{1}{c_{l_3}-l_{l_3}} + \frac{1}{c_{l_7}-l_{l_7}} = \frac{1}{5.5} + \frac{1}{5.5} + \frac{1}{8} \approx 0.489$.

If we apply our algorithm with $N = 1$, both S_1 and S_2 will send the whole traffic on their left-hand path, because $\frac{1}{11^2} < \frac{1}{10^2}$ and $\frac{1}{10^2} < \frac{1}{9^2}$ (as we consider the $MeanDelay$ objective function, link metrics are $f'_a(l_a) = \frac{1}{(c_a - l_a)^2}$). Thus $OBJ_{N=1} = \frac{1}{1} + \frac{1}{9} + \frac{1}{9} \approx 1.222$. If $N = 2$, the first stratum will be routed on the left-hand path for both commodities and the second one on the left-hand path for (S_1, D_1) and on the right-hand path for (S_2, D_2), because $\frac{1}{6^2} > \frac{1}{9.5^2}$ and $\frac{1}{9.5^2} < \frac{1}{9^2}$. Thus $OBJ_{N=2} = \frac{1}{6} + \frac{1}{4} + \frac{1}{9} \approx 0.528$, which is already far better than $OBJ_{N=1}$.

When N becomes sufficiently large the strata become infinitesimal. Thus the traffic is routed on the left-hand path for both commodities until the portion of traffic routed for both commodities (α) is such that the left-hand path and the right-hand path for (S_1, D_1) have the same cost, i.e. $\frac{1}{(11-10\alpha)^2} = \frac{1}{(10-\alpha)^2}$. This occurs when $\alpha = \frac{1}{9}$. It is already clear that this routing is not optimal because the optimal routing requires S_2 to send all its traffic on the right-hand path. Now the loads on all the links are such that $c_{l_1} - l_{l_1} = 9.889$, $c_{l_3} - l_{l_3} = 9.889$ and $c_{l_7} - l_{l_7} = 9$. If we continue this reasoning until the whole traffic matrix is routed, we can compute that $lim_{N\to\infty} OBJ_N = OBJ_\infty \approx \frac{1}{5.364} + \frac{1}{5.364} + \frac{1}{8.273} \approx 0.494$ which is at 1% from the optimum.

From this example, we can see why our method is not asymptotically optimal. Our method would be asymptotically optimal (i.e. $lim_{N\to\infty} OBJ_N = OBJ_{opt}$ in this case) if there were only one commodity in the network. This is not the case if there are multiple commodities in the network, as we have seen in our example. The non-optimality is due to the non-reevaluation of the paths that were computed in previous steps of the algorithm and whose costs have been increased due to other commodities. Indeed, in our example, if we could change the paths of the first α portion of traffic from S_2 to D_2 when we realize that it is not optimal anymore, we could reach the optimum. Anyway, as we see in our simulations, our algorithm behaves very well in practice.

5.6 Simulations

We have tried our algorithm on two real networks: the operational network of chapter 3 but also Abilene network. We have run our simulation on two traffic matrices per topology: one "low" traffic matrix measured during the night and one "high" traffic matrix measured during peak hours.

Figures 5.2 to 5.5 present the values of the objective functions when N goes from 1 to 20. The values are relative to the optimal value of the objective function computed with an LP solver ($\frac{OBJ_N}{OBJ_{OPT}}$). Be aware that the vertical

scale is not the same for all the graphs. $MinHop$ and $InvCap$ are not on the figures because obviously these have a value of 1 $\forall N$. Results are presented on figures 5.2 and 5.3 for Abilene network and on figures 5.4 and 5.5 for the operational network. We see that in many cases, we do not have to divide the traffic matrix into many strata to obtain a routing scheme close to the optimum.

Please note that the optimal routing for each objective function is different. So it is impossible to compare two points of two different objective functions on the figures presented in this section. Only two points of the same objective function can be compared.

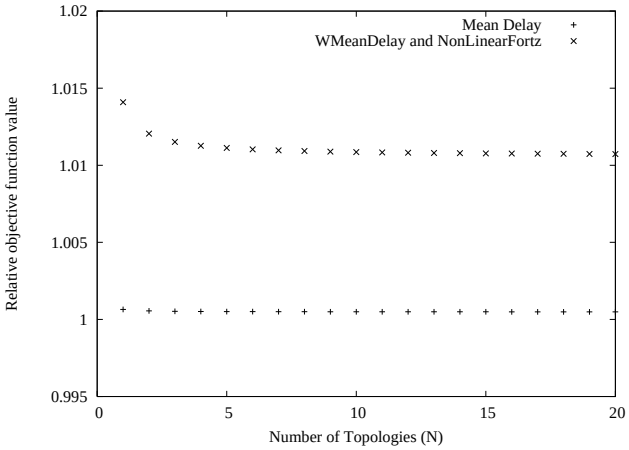

Figure 5.2: Abilene Topology ("low" TM)

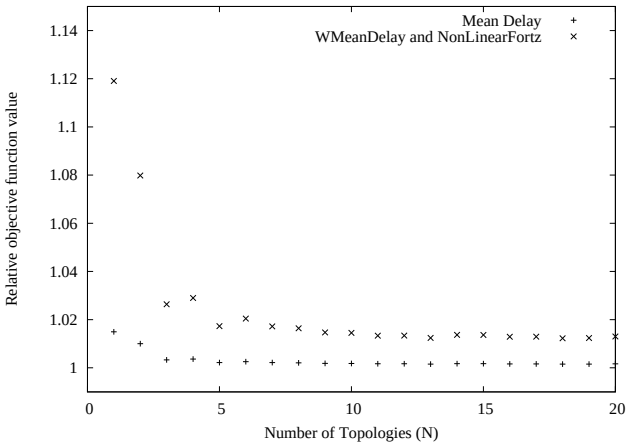

Figure 5.3: Abilene Topology ("high" TM)

If we agree to have a precision ϵ of 1.8%, we can use only $N = 1$ for both networks for low loads (low TMs). For high loads (high TMs), we have to use $N = 5$ to achieve this precision on Abilene network, or $N = 3$ on the other operational network. Depending on the chosen objective function, desired precision can be obtained with lower N value.

On Abilene Topology, $WMeanDelay$ and $NonLinearFortz$ provide exactly the same relative values because all the links of this topology have the same capacities. Thus $NonLinearFortz = \sum_{a \in A} \frac{l_a}{1-\frac{l_a}{c_a}} = \sum_{a \in A} \frac{c_a.l_a}{c_a - l_a} = c \times \sum_{a \in A} \frac{l_a}{c-l_a} = c \times WMeanDelay$

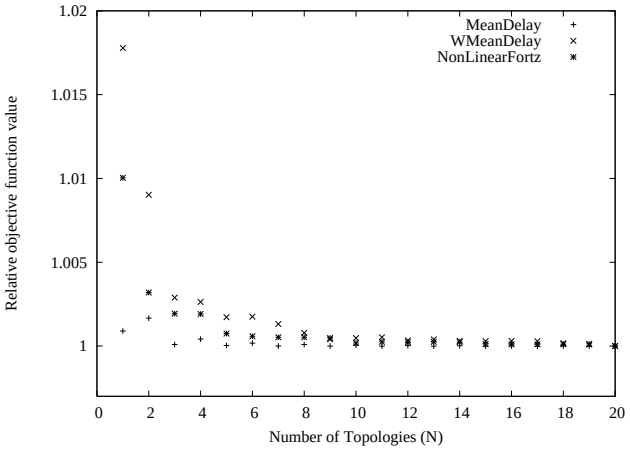

Figure 5.4: Operational network Topology ("low" TM)

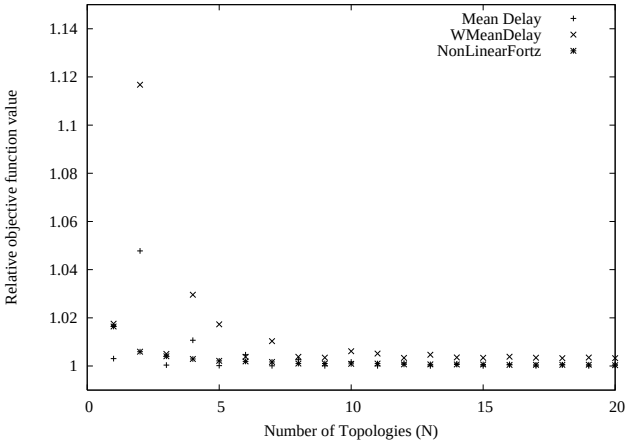

Figure 5.5: Operational network Topology ("high" TM)

On figures 5.2 and 5.3 (Abilene network), we can see that there is a gap of about 1% between our solution and the optimum value for $WMeanDelay$ and $NonLinearFortz$[6], while there is almost no gap for $MeanDelay$ objective function. On figures 5.4 and 5.5 (operational network), there is almost no gap for any objective function. This highlights that the gap that may be observed depends on the topology and the traffic matrix. For information, the gap is also

[6]This means that in this case our algorithm converges to a solution which is at about 1 % of the optimum.

Size of the network		Computation time (in sec)	
Nb Nodes	Nb Links	Our method	LP method
11	14	0.549	0.248
~ 20	~ 40	6.563	23.251
30	60	15.475	634.64
35	70	27.410	1322.586

Table 5.2: Computation time of our method compared to LP formulation

at most 1% on the other topologies of section 5.7. We think this is a quite low value.

5.7 Efficiency

We have used an LP formulation of the routing problem (as in chapter 4) to find the optimal routing scheme so that it was possible to measure the gap between our algorithm and the optimum in preceding section. In this LP formulation, we had to linearize our non-linear objective functions[7]. We have to choose the number of linear pieces of the approximation. Increasing the number of linear pieces in the approximation will increase the quality of the solution found, but it will also increase the running time. Figure 5.6 shows the precision of LP formulation when we increase the number of lines in the piecewise approximation. We present on this figure relative values to the minimum observed over all tests (in this case the minimum occurs when 51 linear pieces are used). We can see that when more than 30 pieces are used the error is at most 0.02%, which is considered as very good. On figure 5.7 we can see the computation time when the number of pieces increase. We can notice that the computation time does not increase that much when we increase the number of linear pieces in the approximation. In fact the big problem of LP formulations is the size of the network. Table 5.2 shows the computation time of the LP solver compared to our method on Abilene and the operational network, but also on two other generated networks. The two additional networks are generated using the BRITE topology generator ([MLMB01]) and the traffic matrix using the TOTEM toolbox ([TOT]). The results are presented for 20 linear pieces in the approximation for the LP method and when dividing the Traffic Matrix in 20 strata for our method. All the simulation times of this chapter are measured on an IBM computer eServer 325 with 2 AMD opteron 2GHz 64 bits processors and 2GB of memory. As LP solver, we have used CPLEX 9.1 (64 bits version). We can see on the table that our algorithm could also be used as benchmark instead of LP for large topologies where LP is not usable due to the high running time. Note that CPLEX is the most powerful solver among those we have tested.

[7] As explained in chapter 4.

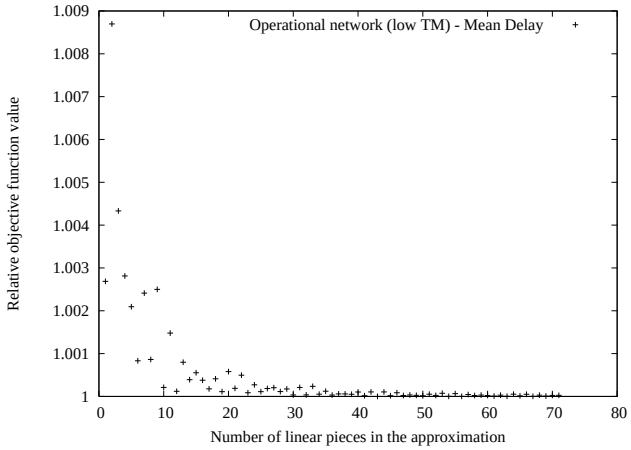

Figure 5.6: Precision of LP routing scheme

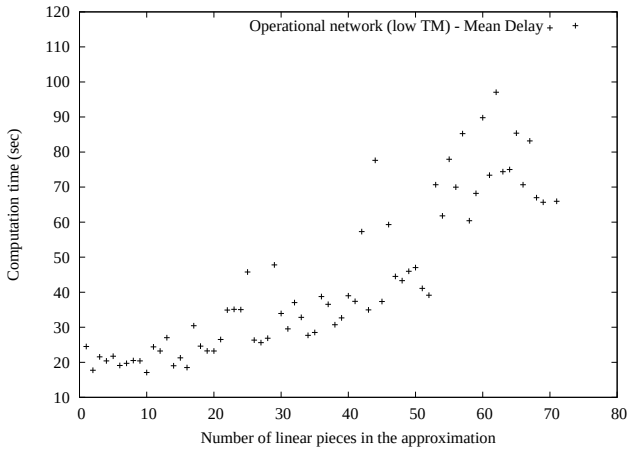

Figure 5.7: Computation time of LP routing scheme

5.8 Implementation on routers

The routing scheme which is found by our method can be implemented on routers using either Multiple Topology Routing ([PSS08, PMR+07]) or multiple MPLS LSP full-meshes[8].

5.8.1 Multiple Topology Routing

In this case, there are two sub-problems. The first problem is to divide the traffic matrix into N sub-matrices. Each router of the network has to map

[8]In [MM05] Menth and Martin propose to use Multiple Topology Routing to provide network resilience, while in [KGST07] Kwong et al. propose to use Multiple Topology Routing to improve service differentiation.

each packet to one of the N topologies. Usually, load balancing is done using a hash function which is based on an identifier of the flow so that all the packets of a flow are forwarded along the same path, thus avoiding packet reordering. The flow id is usually composed of five fields: source and destination addresses, source and destination ports and protocol number.

In our case, the hash function has to be the same in all the routers of the network to avoid cycles. Indeed cycles could appear if one node associates one packet with one topology and the following node associates this same packet with another one. The hash function can be $mod(flow\_id, N)$, for example.

The second problem is to find the N sets of metrics. In our case it is simple. The N sets of metrics are the values of the first derivatives at each step of our algorithm.

5.8.2 MPLS full-mesh

It is simpler with MPLS. The paths of the multiple full-mesh are the paths computed at each step when running SPF algorithm on updated metrics. We also have to use a hash function to associate a packet with one of the multiple LSPs available, but in this case, the hash function can be different in each (ingress) router.

With a triple full-mesh, large backbones with 200-300 egress points would require 600-900 LSP heads at an ingress router. Core routers may need an order of magnitude more transit LSPs. These numbers are far below the thousands of LSP heads and tens-of-thousands transit LSPs that equipment vendors can support today.

5.9 Conclusion

Our algorithm provides a good way to approach the optimal routing scheme for an objective function which is of the form of $\sum_{a \in A} f_a(l_a)$ and for which $f_a(x)$ is convex. To approach the optimal routing scheme, we divide the traffic matrix into N equal strata. For any chosen N, we have proposed two methods to implement corresponding routing scheme in the routers: Multi-Topology Routing or MPLS multiple full-mesh. We have highlighted the trade-off between low N and good precision. Furthermore, our algorithm can be used on large topologies to compute a near-optimal benchmark solution where LP solvers are inefficient. For these topologies, the near-optimal value found by our algorithm can be used to estimate the quality of a heuristic routing scheme. Moreover, the source code of this algorithm is freely available in the TOTEM Toolbox[TOT], so using it does not require any expensive license as it is the case for professional LP solvers like CPLEX.

6

Combined Intra- and Inter-domain Traffic Engineering using Hot-Potato Aware Link Weight Optimization

As presented in chapter 2 the classical approach to intradomain traffic engineering in OSPF/ISIS networks consists in finding the set of link weights that minimizes a network-wide objective function for a given intradomain traffic matrix. We will see that this approach is inadequate because it ignores a potential impact on interdomain routing. Indeed, the resulting set of link weights may trigger BGP to change the BGP next hop for some destination prefixes, to enforce hot-potato routing policies. In turn, this results in changes in the intradomain traffic matrix that have not been anticipated by the link weight optimizer, possibly leading to degraded network performance.

In this chapter we propose a BGP-aware link weight optimization method that takes these effects into account, and even turns them into an advantage. This method uses the interdomain traffic matrix and other available BGP data, to extend the intradomain topology with external virtual nodes and links, on which all the well-tuned heuristics of a classical link weights optimizer can be applied. A key innovative asset of our method is its ability to also optimize the traffic on the interdomain peering links. We show, using our operational network dataset as a case study, that our approach does so efficiently at almost no extra computational cost.

6.1 Introduction & Motivation

In OSPF/ISIS networks, the only way to optimize the traffic is by finding an appropriate set of link weights that minimizes a given domain-wide objective function. In its simplest form the resolution of this optimization problem needs to take as inputs (1) the network topology with unknown link weights, (2) the chosen network-wide objective function, and (3) an intradomain traffic matrix, which specifies the amount of traffic between every pair of ingress/egress nodes.

However this approach is unaware of the interdependence between intradomain and interdomain routings. Actually the real traffic demand is an interdomain traffic matrix (from IP prefix to IP prefix), while the intradomain traffic matrix (from ingress to egress nodes) is only the result of applying BGP routing decisions on the interdomain traffic matrix (TM). Even if we consider that the interdomain TM and the interdomain (BGP) routes are invariant, the intradomain TM may still vary if some link weights are changed inside the domain. This is due to the so-called hot-potato (or early exit) decision rule implemented by BGP.

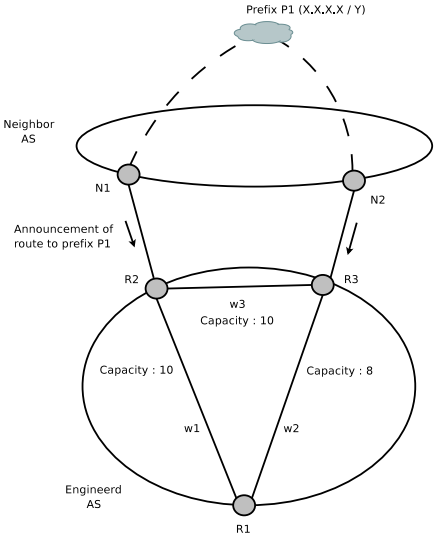

Figure 6.1: Toy Example

The toy example depicted in figure 6.1 suffices to illustrate the problem[1]. This figure shows a domain with three nodes: an ingress node R_1 possibly sending traffic to egress nodes R_2 and R_3, and three intradomain links of weights w_1, w_2 and w_3. Suppose this AS has two peering links (respectively R_2-N_1 and R_3-N_2) with a neighboring AS providing connectivity to the IP prefix P_1. Further suppose that no BGP rule of higher precedence than the hot-potato rule has been able to make a selection between R_2 and R_3. If the link weights are inversely proportional to the link capacities shown on the figure, then ingress node R_1 will choose to reach this prefix through egress node R_2 according to the hot-potato rule (because $w_1 = 1/10 < w_2 = 1/8$). If R_1 has 5 units of traffic to send to P_1, then the intradomain TM is just 5 units from R_1 to R_2 and no traffic elsewhere.

Now suppose that we run a link weight optimizer that tries to minimize the maximum link utilization, while allowing equal cost multipath (ECMP)[Moy98]. A possible optimal link weights setting is $w_1 = 2$, $w_2 = w_3 = 1$, leading to two IGP equal cost paths from R_1 to R_2 and to a maximum link utilization of 2.5/8 on link R_1-R_3. However, if the weights are set as proposed, the hot-potato rule will now select R_3 as egress node to reach P_1 (because $w_2 < w_1$), and the

[1]This example network is similar to the one used in [CEDFQ06]

resulting intradomain TM is actually 5 units of traffic from R_1 to R_3, with a maximum utilization of 5/8 on link R_1-R_3. Clearly, the outcome is much worse than expected, and even worse than keeping the initial weights setting!

This toy example illustrates that we cannot rely on the intradomain TM to solve the optimization problem, because it is not invariant under link weights changes, possibly leading to degraded network performance.

Even though this toy example is not representative of real networks with real traffic, we will show in section 6.5, by using our operational network dataset as a case study, that this phenomenon can really happen with bad consequences, because a substantial amount of prefixes/traffic may be subject to hot-potato (re)routing. For the case study in section 6.5 we show that 97.2% of the prefixes have multiple possible egress points, which amounts to 35.6% of the traffic on average. Without taking hot-potato effects into account, we will show that the link weights proposed by a classical LWO may result in link utilizations close to and even above 100%, while the tool expected maximal link utilizations of only about 35%.

We propose a link weight optimization method that takes these hot-potato effects into account, and even turns them into an advantage. To this end we use as inputs the (hot-potato invariant) interdomain TM and some BGP data, both collected inside the domain, to infer the set of IP prefixes that can be reached by at least two egress nodes and for which no BGP rule of higher precedence than the hot-potato rule has been able to make a selection, i.e. for which the hot-potato rule can potentially be the tie-breaker. We call this subset the hot-potato prefixes, and from now on in this introduction we will only consider these prefixes.

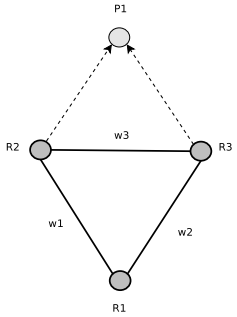

Figure 6.2: Toy Example - Simplified Version

Our method is based on an extension of the intradomain topology with external virtual nodes and links. A first naive and unscalable way to solve the problem would consist in adding a virtual node per hot-potato prefix and attach this node to all possible BGP next-hops for this prefix. This is depicted on figure 6.2 for the toy example of figure 6.1. If we now run LWO on this virtual topology, while still allowing equal cost multipaths, including multiple BGP next-hops, an optimal weights setting is $w_1 = w_2 = w_3 = 1$, which will split the 5 units of traffic evenly on the two paths R_1-R_2-N_1 and R_1-R_3-N_2.

However, the number of hot-potato prefixes can be very large and we would like to keep the number of virtual nodes roughly similar to the number of or-

dinary nodes. To this end we propose to aggregate all virtual nodes attached to exactly the same sets of BGP next-hops. They are indeed indistinguishable with respect to intradomain routing. On the operational network that we have considered, the number of such nodes boils down from 160,000 to only 26. We further show by considering the amount of traffic sent to these aggregates, that we can reduce this set to only 5 virtual nodes without neglecting more than 0.06% of the total "hot-potato" traffic.

An asset of our method lies in reusing the well-tuned LWO heuristics on this extended topology. Moreover, we have also extended this intradomain traffic engineering problem to the peering links, by taking these links into account in the objective function. In our simulations this method allowed us to reduce the maximal interdomain link utilization from 70.1% to 36.5%.

In section 6.2 we review related works. In section 6.3 we present iBGP multipath load sharing, an optional feature of BGP that we use in this chapter. In section 6.4 we formulate the problem and propose our BGP-aware LWO. In section 6.5 we show an application of the method using an operational network. In section 6.6, we discuss future work concerning possible control of incoming interdomain traffic and also potential oscillations. Finally, section 6.7 concludes this work.

6.2 Related Work

The first LWO algorithm proposed by Fortz et al. in [FT00] is based on a tabu-search metaheuristic and finds a nearly-optimal set of link weights that minimizes a particular objective function, namely the sum over all links of a convex function of the link loads and/or utilizations. This problem has later been generalized to take several traffic matrices [FT02] and some link failures [FT03] into account. In our LWO we reuse the heuristic detailed in [FT00], but we have adapted this algorithm to consider the effect of hot-potato routing. All the later improvements to this algorithm (i.e. multiple traffic matrices, link failures) could be integrated in our new LWO in a similar way.

The fact that the intradomain TM is not the correct input for many Traffic Engineering problems had already been pointed out in [FGL+01, FGL+00] by Feldmann et al., who suggested to consider the set of possible egress links in the traffic matrix. In [Rex06] several extensions to the classical LWO problem are briefly described by Rexford, including a sketch of a method that resembles ours. Our work is in line with this recommendation, as we connect several equivalent egress nodes to a single virtual node representing the destination, but we propose a complete method to solve the link weight optimization problem, applicable to intradomain and peering links, and we demonstrate its efficiency on an operational network. In [ANB05] Agarwal et al. study how hot-potato routing influences the selection of IGP link weights and how traffic to neighboring ASes shifts due to changes in the local AS's link weights. In their measurement study they find that weights resulting from ignoring hot-potato interactions can be sub-optimal by as much as 20% of link utilization. We show in this paper that the sub-optimality can be much larger. They also find that as much as 25% of traffic to a neighboring AS can shift the exit point due to a local AS IGP link

weight optimization. They have developed a patch to their link weight optimizer which recomputes the intradomain traffic matrix from the interdomain one at each step of the optimization. Their optimizer does not consider directly the interdomain traffic matrix. Also, their link weight optimizer does not engineer interdomain links and they have tested their algorithm on only 80% of the total traffic of their private ISP while we have tested it on 99% of traffic of an operational network. Finally their source code is not available, while our algorithm is available in open-source in the TOTEM toolbox.

Cerav-Erbas et al. have already shown in [CEDFQ06] that the link weights found by a LWO may change the intradomain TM considered as input. In that paper they also show that applying LWO recursively on the resulting intradomain TM may not converge. They propose a method that keeps track of the series of resulting TMs and at each iteration they optimize the weights for *all* the previous resulting intradomain TMs simultaneously. However, they do not consider the general problem with multiple exit points for each destination prefix, let alone taking advantage of it.

In [WXQ+06] Wang et al. propose to take the interdomain routing into account by splitting the problem into two subproblems. The first one consists in optimizing the mapping of every (hot-potato) destination prefix to a single egress point. This can then be implemented in BGP by assigning a higher local preference to the route received by the chosen egress node. The second subproblem is then the classical link weight optimization for the resulting (and now invariant) intradomain TM. In our approach we solve both subproblems in one step with the usual LWO and we do not need to assign local preference values to pin down every destination prefix to a unique BGP next-hop. By keeping all the potential next-hops we have more flexibility to engineer the network.

Several studies have shown that the proportion of IP prefixes whose next hop is selected by the hot-potato criterion can be very large in ISP networks. Based on measurements of one ISP network (AT&T's tier-1 backbone network) Teixeira et al. show in [TSGR04] that hot-potato routing changes are responsible for a big part of BGP routing changes. While this is not the main goal of that paper they have measured that more than 60% of the prefixes can be affected by the hot-potato routing changes and that these *hot-potato* prefixes account for 5-35% of the traffic in the network. It is also explained in [TDRR05] that *Since large ISPs typically peer with each other in multiple locations, the hot-potato tie-breaking step almost always drives the final routing decision for destinations learned from peers, although this is much less common for destinations advertised by customers.* The authors show that although most routing changes do not cause important traffic shifts, routing is a major contributor to large traffic variations. This demonstrates that it is very important to take BGP routing considerations into account when running traffic engineering algorithms.

In [RTZ03] Roughan et al. analyze the effects of imprecision in the traffic matrix due to estimation techniques on traffic engineering algorithms. While the effects of these imprecisions seem to be quite limited, we show in this work that the effects due to hot-potato routing can be very large. This is an important result as this highlights that not taking hot-potato effects into account cannot be simply seen as resulting in little (harmless) imprecision in the traffic matrix. Hot-potato errors in the TM can really be a big problem for intradomain TM-

based TE algorithms optimizing the link weights.

To the best of our knowledge this work is the first algorithm to find the best possible set of link weights to engineer intra- and inter-domain links while taking hot-potato effects into account.

6.3 Splitting the traffic among multiple paths

In this chapter we are particularly interested in routes that are selected using the 6th criterion of the BGP decision process presented in section 1.1.2, which refers to the link weights of the domain to select the best route toward a destination (i.e. Hot-Potato criterion).

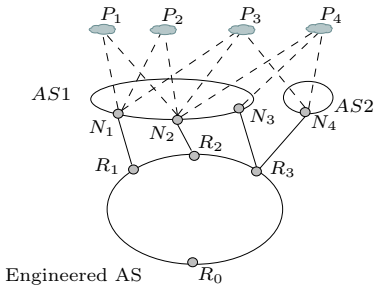

Figure 6.3: Example Topology

Consider the network of figure 6.3. Suppose that routes to P_1 are announced by N_1 to R_1 and N_2 to R_2 on eBGP sessions. Suppose that the routes announced by these two routers have the same attributes (i.e. local-preference, AS-path, origin number and MED) after passing the input filters of routers R_1 and R_2 (this is very frequent in practice for routes that are received from the same neighboring AS). Suppose also that these two routes are forwarded by R_1 and R_2 to R_0 on iBGP sessions. Usually the attributes are not changed when forwarding routes on iBGP sessions. So R_0 has two routes to reach P_1 and these two routes are equivalent w.r.t. criteria 1 to 4. Both are received on iBGP sessions so are also equivalent w.r.t. the 5th criterion. In this case R_0 will use its IGP distance to R_1 and R_2 to select the best route toward P_1. We say that this route is chosen using the hot-potato criterion by router R_0. Note that R_1 and R_2 will directly forward traffic toward this prefix on their interdomain link using the eBGP>iBGP criterion (the 5th criterion of the BGP decision process presented in section 1.1.2). So we see that prefixes that are routed via the hot-potato criterion by some routers will be routed according to the eBGP>iBGP criterion by some others and vice-versa.

Now if R_1 and R_2 are at the same distance from R_0, the 7th criterion will be used. By default only one next hop can be chosen and a tie-break selects the best route. But it is also possible to enable iBGP multipath load sharing ([BGP, Fou]) to balance the load on both paths. As for intradomain ECMP, a hash table is used to select the particular route of a packet. Figure 6.4 supposes that iBGP multipath is activated and that R_1 and R_2 are at the same distance from R_0. In this case the traffic going from R_0 to P_1 will be split evenly on both paths.

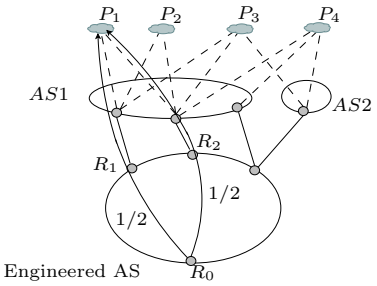

Figure 6.4: iBGP multipath

If both ECMP and iBGP multipath are activated, we have to clarify how the traffic is split between multiple paths. Consider figure 6.5. Suppose that R_1 and R_2 are at equal distance from R_0. Two equal cost paths are available from R_0 to R_1 and only one from R_0 to R_2. The load sharing implementations in routers we are aware of will send 1/3 of the traffic on each of the 3 available paths at router R_0.

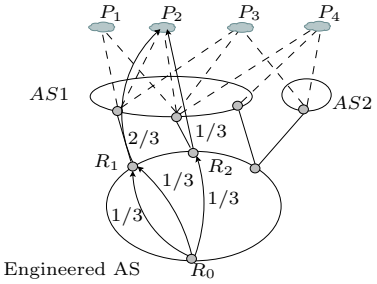

Figure 6.5: ECMP + iBGP multipath

6.4 A BGP-aware link weight optimizer

In this section we present our model of the general traffic engineering problem. We will use the network of figure 6.3 to illustrate all the presented concepts.

6.4.1 Formulation of the extended traffic engineering problem

A network is modeled as a directed graph, $G = (N, L)$ whose vertices and edges represent nodes and links. The basic intradomain topology is composed of all the nodes and links that belong to the AS. We consider two disjoint categories of destination prefixes. The *single-egress prefixes* are those prefixes for which the BGP next-hop is chosen by one of the first 4 BGP criteria. The *hot-potato prefixes* are all the other prefixes. For each of them there is at least one router in the domain that has used the hot-potato criterion, or a following one, to select the next-hop. For each of these *hot-potato prefixes* however, there are also at

least two other routers that forward traffic according to the 5th BGP criterion (eBGP>iBGP), that has precedence over the hot-potato criterion (as shown in the example of section 6.3). The traffic forwarded to the *single-egress prefixes* constitutes a (hot-potato invariant) intradomain TM, called TM_{invar}. We also include in that TM_{invar} the traffic forwarded to the *hot-potato prefixes* originated from the particular nodes that uses the 5th BGP criterion (eBGP>iBGP) to choose their best route. The remaining traffic forwarded to *hot-potato prefixes* constitutes TM_{hp}.

For every hot-potato prefix we conceptually add a *virtual node* representing it. Then for every peering link on which equivalent BGP routes (up to criterion 4) have been announced for that prefix, we extend the intradomain topology with a link+node pair representing this peering link and the neighboring router on the other side of this link. Finally we attach all these neighboring routers to the virtual node (representing the hot-potato prefix) by adding *virtual links*.

Therefore we have three disjoint sets of edges in the topology: L_{intra} is the set of intradomain links, L_{inter} is the set of interdomain links, and $L_{virtual}$ is the set of virtual links. Similarly we split the nodes in the topology into three disjoint sets: N_{intra} is the set of routers from the local AS, N_{neigh} is the set of border routers in neighboring ASes, and $N_{virtual}$ is the set of virtual nodes.

Figure 6.6 shows such a topology. It is the same as figure 6.3 where prefixes are replaced by virtual nodes and possible paths to prefixes are replaced by virtual links. P_1, P_2, P_3 and P_4 are HP prefixes that compose $N_{virtual}$. The BGP-equivalent routes (up to rule 4) are announced by N_1 and N_2 for P_1 and P_2, by N_1, N_2 and N_4 for P_3, and by N_2, N_3 and N_4 for P_4. $L_{inter} = \{R_1 - N_1, R_2 - N_2, R_3 - N_3, R_3 - N_4\}$ and $L_{virtual} = \{N_1 - P_1, N_1 - P_2, ...\}$. $N_{intra} = \{R_*\}$, $N_{neigh} = \{N_*\}$, and $N_{virtual} = \{P_*\}$.

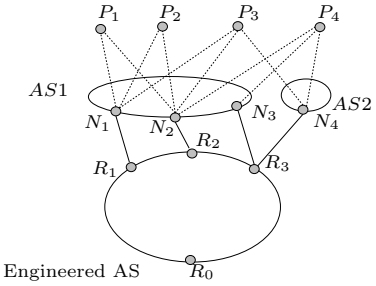

Figure 6.6: More Complex Topology with virtual nodes

Each virtual link ($l \in L_{virtual}$) has infinite capacity $c_l = \infty$ and a fixed weight $w_l = 0$. Every other link ($l \in L_{intra} \cup L_{inter}$) has a capacity c_l and a weight w_l. Let us note that interdomain and virtual links are directed (toward the destination prefix) as no transit via a virtual node is allowed.

The traffic will follow the shortest path(s) based on the link weights. If there are multiple equal cost paths, traffic is considered to be evenly split among them, as shown on figures 6.4 and 6.5.

Once the paths are chosen, we can associate with each link l a load l_l, which is the proportion of traffic that traverses link l summed over all pairs of source/destination nodes. The utilization of a link l is $u_l = l_l/c_l$.

The goal of the LWO is then to find the set of link weights that minimizes our network-wide objective function based on the loads and/or utilizations of intradomain and interdomain links.

6.4.2 Aggregating prefixes

The problem as formulated in the preceding section is not solvable in practice. Indeed the number of prefixes in the BGP routing table of an internet router is about 160,000[2] and so in the worst case all the prefixes are hot-potato prefixes and about 160,000 nodes would be added to the intradomain topology (see section 6.5 for the actual number of hot-potato prefixes in the operational network we have studied). However all prefixes that are reachable through exactly the same set of possible nodes $\in N_{neigh}$ can be aggregated (e.g., nodes P_1 and P_2 in figure 6.6 can be merged) as they are indistinguishable from an intradomain routing perspective. This will drastically reduce the number of virtual nodes. Note that if n is the number of peering links of the AS, there can still be 2^n virtual nodes in the worst case. In practice however it is much lower, as explained in [FBR03]. Indeed routes are often announced with the same parameters on peering links with the same neighbor AS. For the operational network we have used as a case study, the number of peering links traversed by hot-potato traffic is 18. Out of 2^{18} possible different combinations of peering links, only 26 are actually observed.

We can still go one step further by taking the traffic destined for each aggregated virtual node into account. For example, in our case study we have noticed that no traffic is sent to 8 of them, and only a very small volume of traffic is sent to 13 others, thus leading to 5 nodes receiving 99.94% of the hot-potato traffic (TM_{hp}). So we can basically extend the intradomain topology with these 5 virtual nodes without really losing accuracy. This is really significant for the practical efficiency of the LWO. More precisely, using 5 nodes instead of 18 reduced the average computation time of the algorithm from 582 to 140 seconds[3] without decreasing the quality of the provided solutions. Stated otherwise, the same computational budget would allow us to find a better solution (using more iterations) on the smaller topology.

Figure 6.7 depicts the structure of the aggregated interdomain traffic matrix, with one row per edge node in N_{intra} and one column per edge node in N_{intra} or in $N_{virtual}$.

To build this aggregated interdomain traffic matrix we proceed as follows. Let (s, p) be the traffic volume from an ingress node ($s \in N_{intra}$) to a destination prefix (p). If p is not a hot-potato prefix (i.e., there is only one possible egress node $t \in N_{intra}$), we add this traffic volume to the pair (s, t) in TM_{invar}. If the prefix p is a hot-potato prefix, we distinguish two subcases. If node s is a possible egress node for this prefix, we add this traffic volume to the pair (s, s)

[2]It was about 160,000 when we did collect the dataset. At the time of writing (June 2008) it is more than 255,000 and it continues to increase (source: RIPE routing table report).

[3]This is the average computation time over 14 runs on different TMs with 50 iterations per run. We have used 50 iterations because we have noticed that increasing this number did not significantly improve the quality of the solution found on this data. These simulation times are measured on an IBM computer eServer 325 with 2 AMD opteron 2GHz 64 bits processors and 2GB of memory.

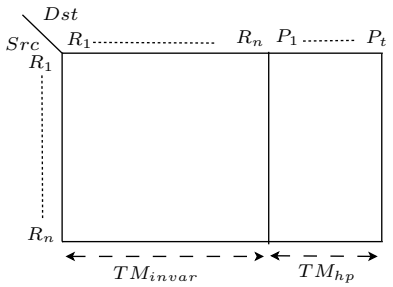

Figure 6.7: The Aggregated Interdomain Traffic Matrix

in TM_{invar} (indeed this traffic will be routed using the eBGP>iBGP criterion). On the other hand, if s is not one of the possible egress nodes for p, we add this amount of traffic to the pair (s, P_i) in TM_{hp}, where $P_i \in N_{virtual}$ is the virtual node associated with the prefix aggregate comprising p. Now we have our aggregated interdomain traffic matrix, which is composed of TM_{invar} and TM_{hp}.

6.4.3 Engineering intra- and interdomain links

In this work we reuse *fortz* objective function which has been presented in chapter 4. It is a piecewise linear convex function of the link utilization and capacity (ϕ_l for link l): $\phi = \sum_{l \in L_{intra}} \phi_l$, where L_{intra} is the set of intradomain links. Others objective functions could also be used in the optimizer. As interdomain links are now part of the topology, we can include these links in the objective function. We are flexible with respect to the inclusion of these interdomain links in the objective function by adding a parameter α which determines the relative importance of interdomain links with respect to intradomain ones. The new function is $\phi = \sum_{l \in L_{intra}} \phi_l + \alpha \sum_{l \in L_{inter}} \phi_l$. In section 6.5 we will compare cases where $\alpha = 0$ and $\alpha = 1$. Values of α in between have not been tested as $\alpha = 1$ seemed to be the good compromise in our case. Indeed as shown in section 6.5.2 it was possible to engineer interdomain links without decreasing the efficiency of the intradomain load balance. Note that it could be different in other networks and in this case it would be interesting to test other values of α.

The inclusion of interdomain links in the objective function is a key advantage of our method as it allows the LWO to engineer these interdomain links in addition to intradomain ones. With a classical LWO there is no point in including interdomain links in the topology and engineer them, because the intradomain TM used as input pins down the egress node anyway, thus assigning the same load on the interdomain links irrespective of the link weights. As we relax the constraints on the egress nodes, it seems natural to take advantage of it to also engineer the traffic on interdomain links. With our method it suffices to include these links in the objective function.

6.4.4 Collecting input data for the optimizer

Our LWO needs as input some information about the traffic and also some BGP data. The required traffic information is the traffic volume from every ingress router to every destination prefix. For the BGP information we have to discriminate the hot-potato prefixes from the other ones. For hot-potato prefixes, we need the set of possible BGP next-hops. For other prefixes, we just need the unique BGP next-hop.

We will mainly describe the method we have used in our case study. We had access to daily dumps containing all the routes received by the monitoring station. In other words, the traces contain for each day all the best routes used by all the routers of the network toward every possible destination prefixes.

We distinguish two categories of prefixes:

- The prefixes for which the same route is selected by all the routers as the best route (they will correspond to our earlier definition of single-egress prefixes);
- The prefixes for which at least two routers in the AS have selected different best routes (they will correspond to our earlier definition of hot-potato prefixes).

The first category of prefixes contains all the prefixes for which the best route is selected by one of the first 4 criteria of the BGP process (local preference, AS path, origin number and MED), and the second category contains all the prefixes for which the best route is selected at a later stage (i.e. by the eBGP>iBGP, hot-potato, or tie-break or load-balancing criteria). Indeed suppose that several routes for the same prefix are received on different eBGP sessions. If one router selects its best route by one of the first four criteria, all the other routers will select exactly the same route by the same criterion, because eBGP data are exchanged "as is" on all iBGP sessions and all the routers are part of the iBGP full mesh. On the other hand if there are at least two equivalent routes after the 4th criterion, then each of these routes will be chosen by at least one router according to the 5th criterion (eBGP>iBGP), namely the border router that has received that route on its eBGP session.

So we can deduce that if we see only one route for one prefix in the BGP trace, this means that this prefix is not a hot-potato prefix. If this prefix appears at least twice this means that this prefix is routed by the 5th, 6th or 7th criterion depending on the router. This prefix is anyway a hot-potato prefix, because even though some routers have chosen their best route by the 5th criterion, other routers must have used the 6th or 7th criterion in this case.

6.4.5 Incorporating changes in a classical LWO

We have modified the classical LWO to include BGP considerations. Three types of links (intradomain, interdomain and virtual) are now present in the model. Intradomain links are unchanged. Interdomain links have a finite capacity and a weight. These are considered in the objective function, weighted by the α parameter. Finally virtual links have infinite capacities, are not considered in

the objective function, and have a null weight. After these modifications a classical LWO, equipped with all its heuristics, can be applied on our extended model.

Notice that the classical LWO considers implicitly that it is possible to split the traffic evenly along several equal cost paths. Therefore it will be necessary to enable ECMP in the network to really get the expected performance. This is anyway a very reasonable choice. Moreover, it was shown (in [ICBD04] for the Sprint network) that ECMP improves robustness. In [SMD03] the authors claim that having multiple shortest paths between pairs of routers provides the ability to switch over to another path in case of link failure without overlapping with the previous path of another node, which could have lead to a transient forwarding loop. It is also said that this is useful to reduce the latency for forwarding-plane convergence for IGP routing changes. Similarly to ECMP, we have considered that it is possible to split the traffic evenly along multiple equal shortest-paths up to the virtual node. So to get the expected performance the network administrator will have to enable iBGP multipath load sharing. Enabling iBGP multipath load sharing is again a natural choice for traffic engineering[4].

6.4.6 Respecting the eBGP>iBGP criterion

If next-hop-self[5] is not activated in the network, it is possible to let the optimizer choose weights on interdomain links. This gives more knobs to tune to the LWO, in addition to the intradomain links weights. The pros is that the LWO may potentially find a better solution, and the cons is the larger search space that increases the computation time to performance ratio. In large networks it may become too costly to assign link weights to interdomain links.

Moreover, assigning weights to interdomain links may contradict the eBGP> iBGP criterion. We explain this point on the simplified network of figure 6.8. Suppose that the LWO has found the link weights indicated on the figure. We can easily compute that the shortest path tree toward destination prefix P_1 is R_1 - R_3 - R_2 - P_1. And that is exactly what the LWO has considered during its optimization. However traffic sent by R_1 to P_1 will actually follow another path, namely R_1 - R_3 - P_1, because according to the eBGP>iBGP rule, which has precedence over the hot-potato rule, R_3 prefers to forward this traffic directly on its peering link, although the path via R_2 has a lower cost (in terms of weights).

In our simulations we force interdomain link weights to 0, while all intradomain links are constraint to have integer weights ≥ 1, so that this problem is avoided. Indeed for example in the simplified network of figure 6.8 the shortest path from R_3 to P_1 will always be R_3 - P_1 (weight = 0) and never R_3 - R_2 - P_1 (weight ≥ 1). Note that setting all the weights of interdomain links to 0

[4]Notice that a network operator which still does not want to activate iBGP multipath load sharing can use the LWO presented in next chapter (chapter 7). Indeed that LWO can be configured to consider iBGP multipath load sharing or not, contrary to the LWO presented in this chapter, which always consider that iBGP multipath load sharing is activated.

[5]Next-Hop-Self is a configuration of BGP in which a BGP border router set its address as Next Hop before transmitting on iBGP sessions a route received on an eBGP session. If Next-Hop-Self is activated, an ingress router will compute the path cost from itself to the egress while it will compute the path cost from itself to the router in the neighboring AS if it is not the case. In the first case it will not consider the cost of the interdomain link while in the latter case it will consider it.

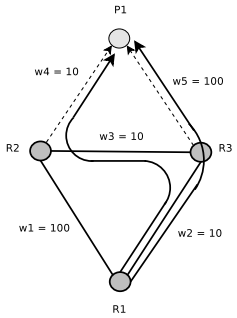

Figure 6.8: Toy Example - with link weights

still allows us to engineer interdomain links by including them in the objective function as explained in section 6.4.3. So this is not a shortcoming and this is confirmed by the good results of the simulation study.

6.4.7 Simplifying the model

When using the LWO without optimizing interdomain links (i.e. only intradomain links are in the objective function), a simplification of the model is possible. Indeed we can remove all the interdomain links (L_{inter}) and all the neighbor nodes (N_{neigh}) from our model. Figure 6.6 would result in this case in figure 6.9 (where P_1 and P_2 have already been aggregated). Indeed in this case the model has just to include all the possible egress nodes for each traffic. This simplification decreases the number of links and nodes of the model and so improves the efficiency of the optimizer.

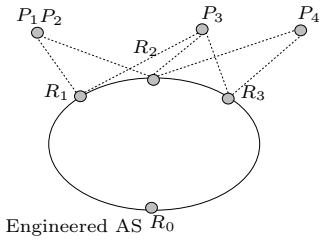

Figure 6.9: Simplified Model

6.5 Simulations on an operational network

We have tested our algorithm on the whole dataset (2,512 aggregated interdomain traffic matrices) presented in chapter 3. For this study it is important to note that it is a transit network that has two providers connected with about 10 interdomain links, has other peer ASes connected with about 15 shared-cost links, and has more than 25 customer ASes, which are mainly single-homed. The total traffic exchanged is about 10 Gbps on average.

In this network there is an iBGP full mesh, MEDs are currently not used, and there are three different local preference values: the lowest value is used for routes learned from provider links, the intermediate value is used for routes learned from shared-cost peering links, and the highest value is used for routes learned from customer links. Route parameters are exchanged unmodified on all iBGP sessions. We have used the technique exposed in section 6.4.4 to build our model.

The average number of prefixes is 160,973 of which 97.2% (156,407) are hot-potato prefixes. If we now take traffic into account, we have measured that these 97.2% amount to 35.6% of the traffic on average. This is still enough to have a significant impact on the link loads of the network. Over all recorded TMs, the peak value is 51.7% of the traffic and the minimal value is 24.6%. Another interesting fact is that on average 99.94% of hot-potato traffic is destined for the 5 biggest clusters of prefixes. The sets of interdomain links giving access to each of these 5 clusters of prefixes are either all peering links to a neighboring AS (for 3 clusters), or a mix of peering links from two such ASes (for 2 clusters).

We have run different versions of the LWO on a large number of traffic matrices. Section 6.5.1 presents some simulation results demonstrating the intradomain traffic engineering capabilities of our algorithm while section 6.5.2 demonstrates that interdomain traffic engineering is also possible. All the simulations consider that ECMP and iBGP multipath are enabled.

6.5.1 Intradomain TE

We first compare a classical LWO (denoted *IntraLWO*) with our BGP-aware optimizer (denoted *BGP-awareLWO*). To execute *IntraLWO* we had to generate for each interdomain TM the corresponding intradomain TM where the hot-potato traffic is routed considering the present (i.e., non engineered) link weights. So these intradomain TMs are those that would be measured in the network. For the comparison we have run both optimizers on all the 2,512 aggregated interdomain TM. Optimizers consider weights in a range from 1 to 150. Figure 6.10 shows the maximal intradomain link utilization (U_{max}) for some worst-case TMs.

We have run *IntraLWO* on every intradomain TM, and computed the resulting maximal link utilization, assuming that the intradomain TM remains invariant (thus ignoring hot-potato effects). In the sequel these values are denoted *IntraLWO-Predicted*. For this link weights setting, if hot-potato effects are taken into account, we get the resulting maximal intradomain link utilization denoted *IntraLWO-Resulting*. These are the real values that would be observed if the optimized link weights were installed in the network. These values are very different, and sometimes the resulting maximal utilization is even worse than the routing without link weight optimization (not present in the figure). Finally we have run our *BGP-awareLWO* and we can see that the maximal link utilizations are very good. Figure 6.10 shows a selection of TMs providing the worst-case values for *IntraLWO-Resulting*[6]. The average reduction of U_{max} from *IntraLWO-Resulting* to *BGP-awareLWO* over all TMs is 4.5%, but let us

[6] In this case we define worst case values as values of traffic matrices providing the highest intradomain maximal link utilizations.

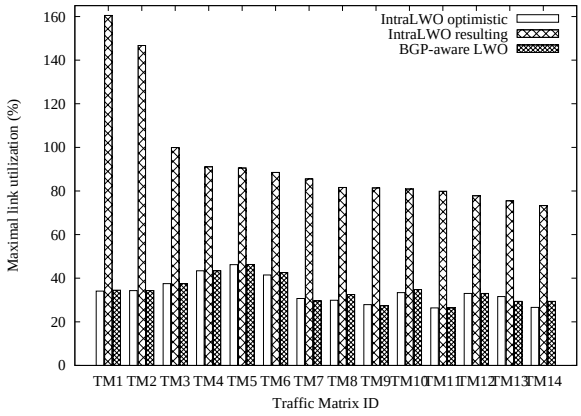

Figure 6.10: U_{max} values for some worst case TMs

outline that the worst-case TMs do matter much more, because the main goal of our LWO is to filter out the unexpectedly bad link weights settings proposed by a classical LWO. In all cases the real minimal value of U_{max} achievable *in practice* are the values of *BGP-awareLWO*, since the *IntraLWO-Predicted* are disqualified in the comparison.

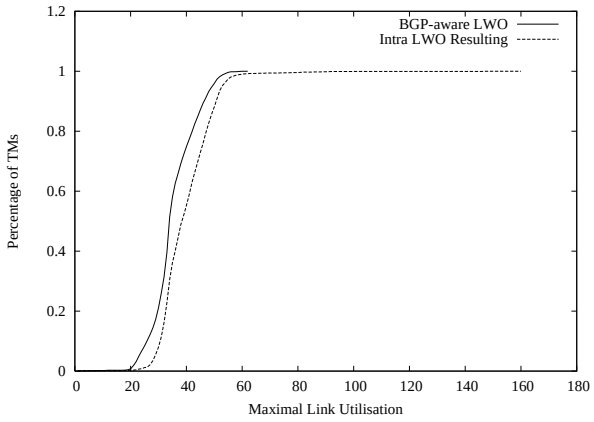

Figure 6.11: CDFs of U_{max} over all TMs for *BGP-awareLWO* and *IntraLWO-Resulting*

Figure 6.11 shows the CDFs (cumulative distribution functions) of the maximal link utilization over the 2,512 TMs for *BGP-awareLWO* and *IntraLWO-Resulting*. *IntraLWO-Predicted* is not depicted on the figure because it would be almost mixed up with *BGP-awareLWO*. We can clearly see that *BGP-awareLWO* is better than *IntraLWO-Resulting*. Figure 6.12 gives the proportions of TMs per range of maximal link utilizations. In this figure we can see that *BGP-awareLWO* takes advantage of the freedom of choice of the egress point(s) for hot-potato traffic. Indeed *BGP-awareLWO* is slightly better than *IntraLWO-Predicted*. For example there are 3.4% less TMs in the [30, 40) range.

This indicates that our optimizer can change the egress point of some hot-potato traffic to better engineer the network.

Concerning the computational efficiency of the LWO, adding the virtual links and nodes has roughly doubled the computation time. We consider that this is not a high cost given the improved quality of the solutions found.

One may wonder why *BGP-awareLWO* does not always find a better solution than *IntraLWO-Predicted* (figure 6.10). It is because the objective function does not strictly minimize the maximal link utilization (i.e., it minimizes the sum over all links of a convex function of the link utilization). Therefore even when the solution is slightly better with respect to the objective function, it can still be a little bit worse with respect to the maximal link utilization.

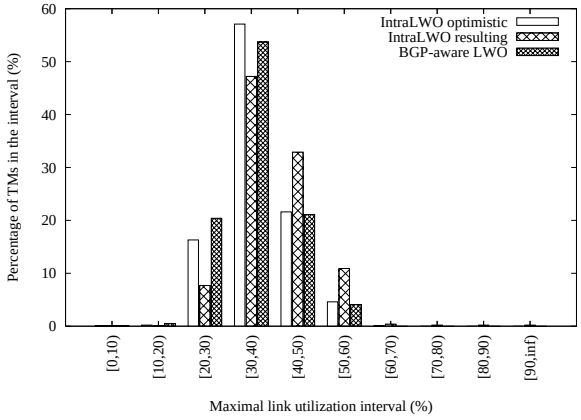

Figure 6.12: Proportions of TMs in each U_{max} interval

6.5.1.1 In-depth analysis of the worst case traffic matrix

In this section we would like to analyze the worst case traffic matrix concerning the maximal link utilization of *IntraLWO-Resulting*.

With the worst case traffic matrix, the maximal link utilization is 160% with the weights optimized with *IntraLWO*. The traffic shifts that happen in this case are depicted on figure 6.13. If $\mathcal{P}_2 < \mathcal{P}_1$[7], traffic on the flow from S to D_1 will be routed on link L, and this will be expected by *IntraLWO*. But if $\mathcal{P}_4 < \mathcal{P}_3$ while before optimization $\mathcal{P}_4 > \mathcal{P}_3$, the hot-potato traffic from S to $VirtualD_4$ will be routed on L and this will NOT be expected by *IntraLWO*. This situation happens four times on the same low capacity link[8] for the worst case traffic matrix, and for quite big hot-potato traffic flows compared to the link capacity.

Before optimization the maximal link utilization is 34.8%. The utilization of the problematic link is only 0.4%. There are only four intradomain shortest paths that use this link and the total traffic on these four flows is 0.6 Mbps.

[7] By $\mathcal{P}_2 < \mathcal{P}_1$ we mean that the sum of the weights of the links of $\mathcal{P}_2$ is smaller than the sum of the weights of the links of $\mathcal{P}_1$.

[8] This link has a capacity of 155 Mbps.

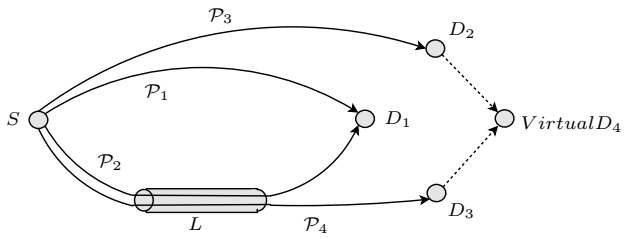

Figure 6.13: Traffic shifts from one shortest path to another

After optimization the problematic link is used in 20 shortest paths instead of 4. This is expected by *IntraLWO* which predicts that these 20 flows will afford 29 Mbps, leading to a utilization of only 18.9% ($< 34.1\%$, *IntraLWO* does not predict that this link is the most utilized link). What is not expected by *IntraLWO* is that 4 of these shortest paths will also attract hot-potato traffic. The hot-potato traffic which is shifted on one of these shortest paths comes from one of the 19 remaining shortest paths using this link. So this shift has no effect on the load of this link. But the hot-potato traffic attracted on the three remaining shortest paths comes from other flows whose shortest path does not include the problematic link. These three flows attract a total amount of 220 Mbps of hot-potato traffic, which is more than the capacity of the link.

6.5.1.2 Increasing the bottleneck links capacities and the traffic matrices

To analyze whether the presence of low capacity links has any impact on our results, we did also run our algorithm on a modified version of the topology, where all the 155Mbps links have been replaced by 622Mbps links. We have also doubled all the elements of the traffic matrices in order to reflect a possible increase in the traffic demand in the future. With this version of the topology and traffic matrices, we have noticed that the impact of hot-potato reroutings on U_{max} after a LWO optimization is larger than with the initial topology and load. Indeed the mean reduction of U_{max} over all TMs from *IntraLWO-Resulting* to *BGP-awareLWO* is now 21.8% instead of 4.5%. This can be observed on the CDF of figure 6.14 for the updated topology and traffic matrices, which should be compared to figure 6.11 for the initial data. We can also observe that for more than 45% of the traffic matrices, *IntraLWO-Resulting* leads to a U_{max} greater than 67.8% which is the U_{max} reached on the worst case TM by *BGP-awareLWO*.

Over all TMs U_{max} have been observed on at least 10 different links. There are 7.5% of the traffic matrices for which U_{max} is greater than 100% for *IntraLWO-Resulting*, and these high U_{max} values can be observed on 6 different links, out of which only 2 are 622 Mbps links. The worst case traffic matrix concerning U_{max} for *IntraLWO-Resulting* induces a utilization of 189.1% on a link whose capacity is 2.5 Gbps. These results demonstrate that it is not always the same lowest capacity link that induces the highest utilization in the network.

We have also analyzed CDF curves for the second, third, fourth and fifth most utilized links. For the second most utilized link, results are similar to

those shown on figure , with a peak maximal utilization for *IntraLWO-Resulting* reaching 175.9%, and a maximal utilization being above 100% for 2% of the traffic matrices. Concerning the third most utilized links, hot-potato reroutings have less disastrous consequences, while still significant in the worst case as the maximal utilization peaks at 95.3% for *IntraLWO-Resulting* while it peaks at 62.3% for *BGP-awareLWO*.

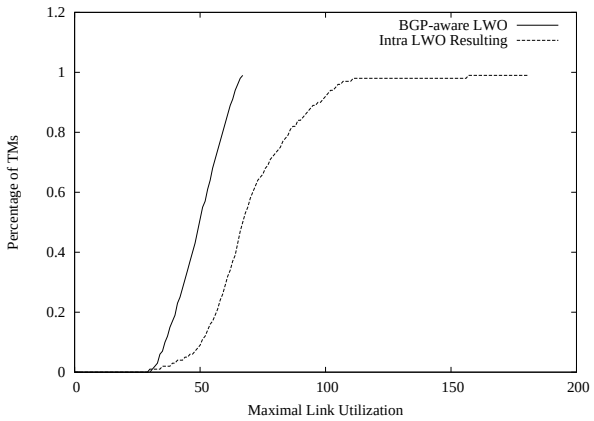

Figure 6.14: CDFs of U_{max} over all TMs for *BGP-awareLWO* and *IntraLWO-Resulting* for the updated topology

6.5.2 Interdomain TE

One of the most innovative feature of our LWO is its ability to engineer traffic on the interdomain links. We first analyze the maximal link utilizations of the interdomain links with the present link weights. The average value of Interdomain U_{max} over all TMs is 36.8%. This value can peak at 73.7%. We have selected the worst TMs in this respect[9] and run *B*GP-aware LWO on them with interdomain links in the objective function. The results are shown in figure 6.15 for the peak TM. The maximal interdomain link utilization is reduced from 73.7% to 36.8% when using *B*GP-aware LWO. It shows that the LWO can take advantage of hot-potato routing to also engineer traffic on interdomain links.

We now show that the optimization of interdomain links is not done at the expense of intradomain links. To this end we have run *BGP-aware LWO* with and without interdomain links in the objective function ($\alpha = 1$ or $\alpha = 0$, see section 6.4.3) on the 50 TMs leading currently to the maximal interdomain link utilization. Figure 6.16 presents the average intradomain and interdomain U_{max} values for these matrices. It shows that *BGP-aware LWO* with all links in its objective function can optimize interdomain links almost without impacting intradomain links. The average intradomain U_{max} value is indeed almost equivalent in both cases.

[9] Here worst case TMs means TMs providing the highest interdomain link utilization with present link weights.

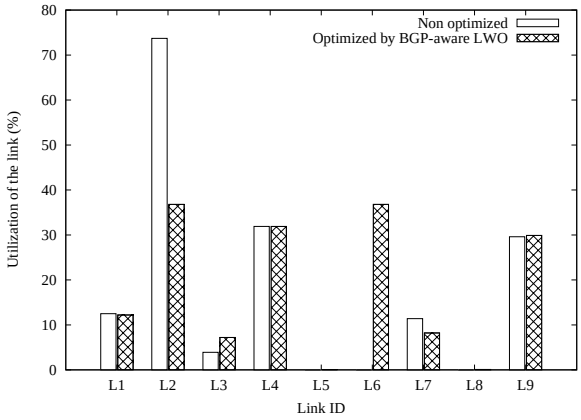

Figure 6.15: Interdomain link utilizations

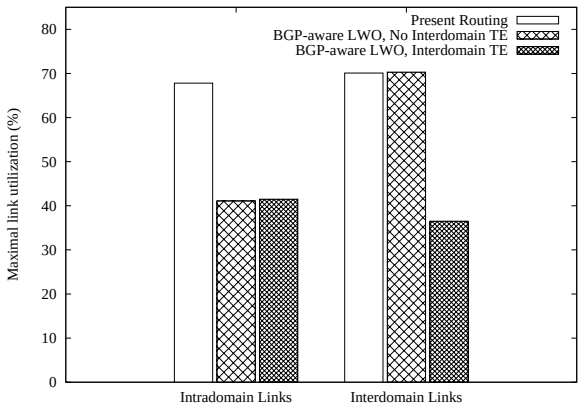

Figure 6.16: Combined Intra- and Interdomain Traffic Engineering

6.5.3 Analysis in a Dynamic Environment

Now we want to analyze the performance of our *BGP-aware LWO* in a dynamic environment where the future traffic matrices are obviously not known. Previous analyses did assume that we were able to measure a traffic matrix that was *representative* of the future traffic matrices. We will now analyze how optimizations based on past traffic measurements do behave when applied on subsequent traffic matrices. We will test different methods to aggregate multiple past matrices into one single traffic matrix which can be used by optimizers. We analyze the efficiency of the global system which is composed of the optimizer, a reoptimization strategy, and an aggregation technique for past traffic matrices.

We will analyze the maximal link utilization (U_{max}), the 90th percentile of link utilizations (U_{per90}[10]) and the mean link utilization (U_{mean}). *BGP-aware LWO* is configured to optimize both intradomain and interdomain links.

6.5.3.1 Routing based on the traffic matrix: with oracle

Figure 6.17 presents the application of BGP-aware LWO on the whole dataset. In this simulation we assume that we are able to predict the actual traffic matrix for the next 15 minutes and that we reoptimize the routes every 15 minutes as well. It is not possible to implement this solution in practice as first we cannot exactly predict the traffic matrix for the next 15 minutes, and also it is not a good idea to reoptimize the routes too often, for stability reasons. This simulation gives us the best routing scheme available with a link weight optimizer on this dataset. We will later see how the routing performance deteriorates when these optimal conditions are not met.

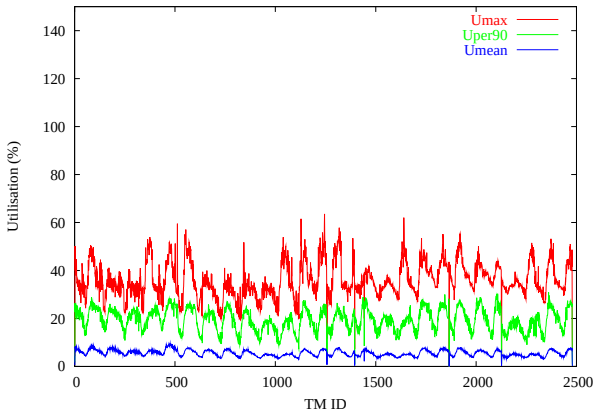

Figure 6.17: *BGP-aware LWO* based on an oracle

6.5.3.2 Dynamic TE

Now we analyze different proposals to optimize the routing scheme based on *past* traffic matrices.

[10]By definition, 90% of the links in the network have a link utilization under U_{per90}.

First we will continue to consider that we are able to change the link weights every 15 minutes (we will call this the reoptimization period).

The simplest technique that can be used to estimate the next traffic matrix is probably to use the last traffic matrix. The corresponding simulation is depicted on figure 6.18. We now see a peak in the maximal link utilization (78.47%) at TM 1444.

We try to figure out the cause of that spike by looking at the traffic and fanout graphs of chapter 3. We can see on figure 3.2 that during TM 1443, the total traffic was going down while on the fanout graphs we have observed that the traffic fanouts were keeping regular values. We do not know the cause of that traffic anomaly. The interesting point is that a link weight optimization based on this non-regular TM has led to a spike in U_{max} when the resulting link weights were applied on the next traffic matrix. This means that using only the information of the last TM to optimize the routes is probably not a good idea.

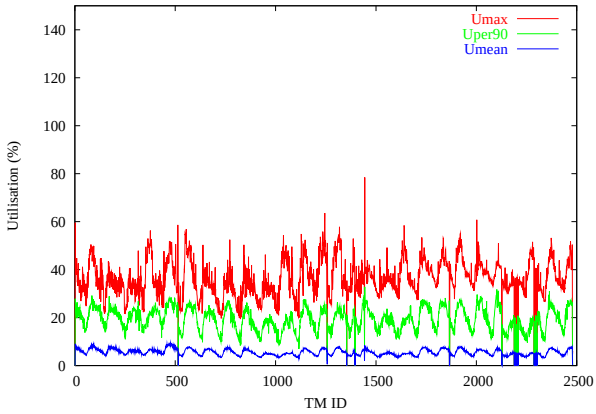

Figure 6.18: *BGP-aware LWO* based on previous TM (the TM computed based on traffic of last 15 minutes)

So we will test another reoptimization technique to keep more information than the last traffic matrix to estimate the next one. We simulate a link weight reoptimization still every 15 minutes, but now each optimization is based on the **maximal** traffic matrix computed on the last 4 traffic matrices (four 15-minute-TMs = one hour). We call this amount of time (i.e. one hour) the memory horizon: the optimizer has only access to the corresponding TMs information.

Note that the maximal traffic matrix is defined so that each *(i,j)* component is the maximal *(i,j)* component for all the traffic matrices within the memory interval[11]. Later, we will also test another method to aggregate multiple TMs into one. These are defined so that each *(i,j)* component of the traffic matrix is set to a value so that X% of the traffic matrices within the memory interval have the *(i,j)* element below this value (and 100-X% are above of course). We will call corresponding TMs the Xth percentile TMs. This aggregation technique should have the advantage to filter out absurdly high values which may be present due to measurements errors and which could lead to erroneous optimization.

[11]The memory interval is defined as: [now-horizon, now].

The resulting simulation (memory horizon of one hour, maximal traffic matrix) is presented in figure 6.19. We can observe that increasing the horizon from 15 minutes to one hour has deteriorated the performance of the system. There are now two peak intervals: between TM 1237 and TM 1258, and between TM 1325 and TM 1327.

In the traffic fanout graphs of chapter 3, we have observed that during period TM 1233 to TM 1257 we have a down period for router R_5 in the source and destination fanouts (figures 3.8 and 3.10). This period matches the first peak period for U_{max}, and so it is probably the cause of that problem. During period TM 1325 to TM 1327 we had observed the starting point (at TM 1325) of multiple abrupt source traffic fanout changes, which is probably due to a source traffic shift from some ingress nodes to others. So this traffic shift was not anticipated by the LWO, which did result in a U_{max} peak until the optimizer is rerun considering this traffic shift.

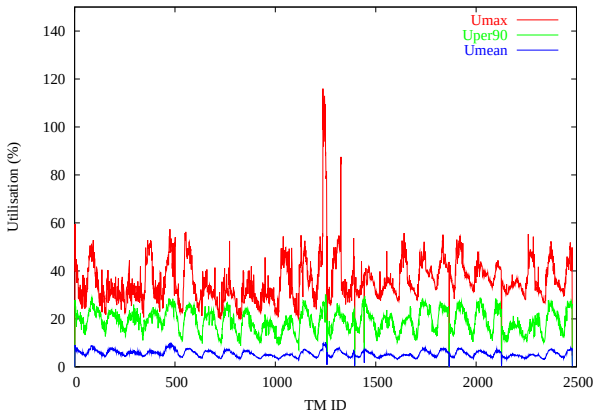

Figure 6.19: Routing scheme reoptimized every 15 min, based on the maximal TM computed over the last hour

Several other configurations that we have also tested are presented in table 6.1. We do not present individual graphs like those presented in figure 6.18 and 6.19 for every analyzed configuration as this will not be very informative. Instead the table presents the following aggregated performance metrics: the **peak** U_{max}, U_{per90} and U_{mean}, but also the **mean** U_{max}, U_{per90} and U_{mean}. We have also included the mean interdomain U_{max}.

We have observed on the utilization graphs (not shown in this work) when they were placed next to the traffic and fanout graphs that only a subset of bad TE performance are related to some traffic changes that can be simply monitored by a network operator (i.e. the source and destination fanouts).

Now we analyze the table. We can observe that some configurations result in very high peak U_{max} values. We can also observe that it is not interesting to have a too short reoptimization period. Indeed many simulations with a short reoptimization period (15 minutes, one hour or even one day) did result in very high peak U_{max}. Moreover short reoptimization periods could create instability in the network, which is not shown in the table.

We can also observe that it is **not** possible to provide a rule such: *"the 90th percentile TM method is always better than the maximal TM technique"*, or *"the longer the memory horizon, the better the performance"*. Indeed we can easily find counter examples in our table.

We have highlighted in *italic* the two lines that seem to be the best configurations for this dataset. The last line of the table shows that we can achieve a good traffic engineering state with one route reoptimization per week. This is probably the best LWO dynamic TE scheme and this is the one we would recommend to use *on this network for this particular dataset*. Of course we cannot deduce general rules for every network from this particular case study, and the trade-off which is good for this network may not be the best on another network. However, on a case-by-case basis, every network could be analysed in a similar way, and particular trade-off, possibly different from this one, could be inferred.

			Intradomain						Interdomain
Reopt.	Memory	Reopt. TM	PEAK			MEAN			MEAN
period	Horizon	method	U_{max}	U_{per90}	U_{mean}	U_{max}	U_{per90}	U_{mean}	U_{max}
15 Min	oracle		63.60	30.89	9.43	35.97	19.38	5.53	35.58
	15 Min		**78.47**	32.82	9.43	36.37	19.40	5.54	35.60
	1 Hour	Max	**115.99**	30.11	10.32	35.93	19.12	5.47	35.43
1 Hour	1 Hour	Max	**112.97**	30.89	10.32	36.07	19.17	5.47	35.61
	1 Hour	75th Perc.	**112.97**	33.39	10.32	36.83	19.30	5.52	35.49
	1 Day	*Max*	*61.54*	*27.61*	*9.49*	*33.60*	*17.25*	*5.24*	*36.95*
	1 Day	90th Perc.	64.95	29.53	9.44	35.00	17.89	5.30	36.43
	1 Day	80th Perc.	**83.55**	29.47	9.87	35.06	18.22	5.33	35.98
	1 Week	Max	**97.75**	27.84	9.60	36.70	16.73	5.24	39.86
	1 Week	90th Perc.	**67.32**	29.05	9.72	35.69	17.49	5.33	38.34
	1 Week	80th Perc.	62.91	28.41	8.48	34.36	17.59	5.26	37.59
1 Day	1 Day	Max	63.41	27.59	8.41	34.09	17.24	5.17	36.83
	1 Day	90th Perc.	64.95	29.53	8.80	35.28	17.62	5.28	35.94
	1 Day	80th Perc.	**69.42**	29.16	8.83	35.18	17.92	5.26	38.09
	1 Week	Max	**97.75**	27.64	8.69	38.10	16.61	5.21	41.21
	1 Week	90th Perc.	**72.69**	30.94	9.91	36.26	17.27	5.29	38.03
	1 Week	80th Perc.	62.91	28.48	9.33	34.09	17.42	5.22	39.15
1 Week	1 Week	Max.	62.91	28.26	8.65	39.18	16.77	5.13	40.47
	1 Week	90th Perc.	63.41	27.77	9.10	36.21	16.89	5.15	38.68
	1 Week	*80th Perc.*	*62.91*	*28.33*	*9.00*	*35.07*	*17.01*	*5.14*	*36.97*

Table 6.1: Peak and mean values of U_{max}, U_{mean} and U_{per90} for various TE routing schemes

6.6 Future Work

6.6.1 Choosing the ingress point for selected traffic

So far we have implicitly assumed that we have no control on the incoming traffic. This is not exactly true. The techniques we could use are for example MEDs, ASPATH prepending or selective prefix advertising. We refer to [QUP+03] for details on these techniques. These techniques can be used to perform INterdomain Ingress Traffic Engineering (INITE) as defined in [GDZ05].

Suppose that we have one client and one provider, and that we learn a route from our client that we announce to our provider. Suppose that we are connected to this provider by two interdomain links. By default we will announce the same route on both peering links. So probably that provider will route traffic toward this prefix according to the hot-potato criterion[12]. Indeed the two routes it has received have exactly the same attributes. We could instead choose the ingress point for traffic coming from this AS for this destination prefix using MEDs or ASPATH prepending. Using MED values require an agreement with our providers.

If we want our LWO to play on ingress points as it plays on egress points, we could extend our model by adding a virtual *incoming* node attached to our two possible ingress nodes with virtual links of infinite capacity. In this case the incoming virtual node models our ability to choose the ingress point for that traffic. There would be one such virtual incoming node per pair of upstream AS and *destination* prefix.

If we associate link weights with the incoming virtual links, we can let the optimizer tune them. Then, if MED attributes are used, the route announced on the eBGP sessions associated with a virtual incoming link will be given a MED value computed as the cost of the path from the incoming virtual node to its associated destination node (i.e. in N_{intra} or $N_{virtual}$). If ASPATH prepending is used, we could prepend every route sent by a potential ingress router for which no traffic is desired.

6.6.2 Potential Oscillation

A known potential issue with LWOs is route instability. As there is no mutual agreement on the egress/ingress points between ASes, it is not guaranteed that two neighboring ASes (say AS_x and AS_y) running their LWO will not oscillate, one reoptimizing its link weights after the other. Indeed each link weights optimization in AS_x can lead to a change of some egress points, changing the traffic matrix in AS_y which may trigger the reoptimization of the link weights in this AS, and so on, leading to route and traffic oscillations.

Such instability may already happen with classical BGP-blind LWOs and, as our BGP-aware LWO does not address this issue, some instability may also potentially exist with our proposition.

In [MWA05] the authors propose a method to negotiate the BGP egress point

[12]This is the case except if this provider has some different input filters for these two routes received at two different points of its network.

between neighboring ASes. This technique should remove oscillations provided that it is possible to fix the egress point, which is not easy in OSPF/ISIS networks. In [MWA05] the authors consider MPLS networks instead.

In [QB05] the authors present another negotiation-based technique to control incoming interdomain traffic of stub ASes. They propose to tunnel the traffic between source and destination cooperating ASes.

The related problem of BGP route oscillations when interdomain traffic engineering techniques are used is considered in [YXW+05], where sufficient conditions are elaborated to guarantee BGP route stability. Unfortunately, these conditions are not fulfilled in presence of LWOs (be they BGP-aware or not), because all LWOs take input traffic into account to choose links weights, which in turn determine egress points for hot-potato prefixes, and thus the corresponding BGP routes.

This problem of potential oscillations is still an open research topic, and was not the primary goal of this work.

6.7 Conclusion

We proposed a BGP-aware Link Weight Optimizer (LWO) that extends the classical (intradomain) LWO to take into account BGP's hot-potato routing principle. The optimized link weights, if deployed, will actually give rise to the link loads expected by the optimizer, contrary to a classical (intradomain) LWO that may lead to unexpectedly high loads on some links when changing weights impacts the intradomain traffic matrix. In practice the method only requires to extend the intradomain topology with a limited number of virtual nodes and links, which preserves scalability, as shown on an operational network used as a case study. The aggregated interdomain traffic matrix associated with this extended topology replaces advantageously the classical intradomain traffic matrix as input to the LWO. On this basis, a classical LWO requires only small modifications to be reused on the extended topology, and this allows us to reuse all its well-tuned heuristics.

The most innovative key asset of the method is its ability to optimize traffic on interdomain peering links as well. We have shown on a case study that it does so very efficiently at almost no extra computational cost, while preserving the 5th BGP routing criterion stating that eBGP-learned routes should be preferred to iBGP-learned ones.

As for a classical LWO, our method can be extended to more general scenarios including several traffic matrices as input and/or possible link failures. Note however that an interdomain traffic matrix used as input is likely to be already more stable (and thus representative) than intradomain matrices. Indeed the interdomain matrix is invariant under all local hot-potato fluctuations, e.g. due to failures. This better stability of the interdomain matrix would allow us to use a smaller set of representative matrices as input, which in turn would give unique link weights settings that are better optimized for each of them.

Even though our method requires additional inputs to build the interdomain traffic matrix and some more computation power, this pays off, because our

BGP-aware LWO clearly outperforms classical (intradomain) LWO.

Then we have analyzed different dynamic routing scenarios. We have shown that it is possible to find a good strategy which does not require too frequent reoptimizations (e.g. once a week).

We have also analyzed the bad performance of some strategies and linked this information with our traffic analysis of chapter 3. We have shown that some bad TE performance are related to some traffic changes that can be simply monitored by a network operator (i.e. the source and destination fanouts). Future works could study how to monitor such simple traffic variables in order to improve the global TE system. We could for example trigger a new optimization when some anomalies are observed. It could also be interesting to give this monitored information to optimizers (e.g. information about a traffic ingress change) which could take this information into account during the optimization, to provide better and more stable routing solutions. We could also imagine to discard measured data that contain *anomalies* to avoid an optimization based on unreliable data, which could lead to bad TE performance.

7

The case of BGP-aware Link Weight Optimizers in ASes using Route Reflectors

The first generation of IGP Link Weight Optimizers (LWOs) was based on presumably invariant intra-domain traffic matrices only, ignoring the fact that updating link weights had a side effect on these traffic matrices due to hot-potato routing, thus resulting in suboptimal link weight settings, and sometimes to very bad performance.

The second generation of IGP LWOs (presented in chapter 6), referred to as BGP-aware LWOs, has been able to optimize link weights while taking hot-potato effects into account. However, these tools relied on the complete visibility assumption fulfilled by e.g. a full-mesh iBGP configuration.

This chapter presents a third generation LWO, still BGP-aware, but also able to work with iBGP configurations based on route reflectors, which usually hide some reachability information from routers. This partial visibility may cause various problems, including path deflections (i.e., the actual egress router is not the expected one), which may in turn create forwarding loops.

Our LWO embeds a BGP routing solver which can always predict the actual egress router, even when route reflectors are used. It can also forbid solutions leading to path deflection. Its efficiency is evaluated on a real dataset, and compared to other LWOs.

7.1 Introduction & Motivation

In chapter 6 we have proposed to integrate the BGP hot-potato rule in a link weight optimizer. But we will see that this method implicitly assumes that every router in the AS is aware of every route announced toward every destination. This is always correct when the iBGP configuration is a full-mesh, but not when route reflectors are used.

Indeed in iBGP full-mesh configurations, each route announcement received by any router on an eBGP session is retransmitted on iBGP sessions to every

other BGP router in the AS. So every router is aware of all the available routes for every destination and it can choose its *global* best route in the whole set of available routes.

The problem of the iBGP full-mesh is that it requires $\frac{n.(n-1)}{2}$ iBGP sessions in an AS composed of n BGP routers. This may be prohibitive in large ASes. To solve this scalability problem, network operators usually install route reflectors, which reduces the number of iBGP sessions ([BCC00]). But it is known that route reflectors can introduce anomalies that are due to partial route visibility. This happens because clients receive only the *best* route from their route reflector. So route reflector clients do not have access to the whole set of available routes. Moreover the route reflector's best route may differ from the client *global* best route (i.e. the route that it would have chosen, had it seen every available route). This can lead to non-optimal hot-potato egress choice. Another anomaly that can be created by route reflectors is the forwarding deflection. This happens when a router selects its best route and on the forwarding path to the egress point corresponding to this best route, there is a router that selects another best route and thus another egress point. In this case we say that the traffic is deflected. We will see an example of network configuration inducing a path deflection in section 7.3.

The biggest problem with deflections is that they can introduce forwarding loops ([GW02]):

- Intra-AS loops due to the combination of multiple deflections in a particular way.

- Inter-AS loops which can appear when incorrect ASPATH information is transmitted by routers for which the traffic has been deflected. As the ASPATH information is not correct, the BGP loop detection mechanism may not work properly.

In this chapter we first want to study the impact of partial visibility on LWOs that make a wrong assumption of complete visibility. This impact can range from a non-optimal traffic engineering solution found by the optimizer to the more dangerous introduction of path deflections in the AS or even forwarding loops. This motivated us to develop a BGP-aware LWO able to take account of partial visibility in presence of Route Reflectors, which allows to detect and forbid path deflections possibly leading to forwarding loops.

The chapter is structured as follows. Section 7.2 presents related works. Section 7.3 analyses in detail the different iBGP configurations that are conflicting with traditional LWOs. Section 7.4 uses simulations to evaluate the impact of partial visibility on LWOs. To this end we run a LWO that wrongly considers complete visibility on a network whose iBGP configuration contains a route reflector. Then section 7.5 presents a new LWO embedding a BGP simulator to predict the actual egress points and detect path deflections. We evaluate the performance of this new LWO in section 7.6 with simulations based on a real dataset. Finally section 7.7 concludes this work.

7.2 Related Work

Section 7.2.1 presents related works on BGP Route Reflector while section 7.2.2 presents related works on LWOs.

7.2.1 Route Reflection

BGP suffers from several problems ([FBR04]). In particular, route reflectors can introduce anomalies that include path deflection or forwarding loops. These problems have been quite extensively studied (in [GW02] for example). In this paper we analyze the impact of these route reflector problems on Link Weight Optimizers. We will see that the partial visibility introduced by route reflectors can have a big impact on the performance of LWOs. But a bigger issue is that LWOs can also introduce path deflections and forwarding loops in a network containing route reflectors. One first simple way to solve these problems would be to ensure complete visibility, but this is really not a trivial task. In [GW02] they show that determination of iBGP configuration correctness is NP-hard[1]. However they provide sufficient conditions on network configurations that guarantee correctness. These sufficient conditions can help but do not solve all problems. First the sufficient conditions only guarantee correctness for one set of link weights. So if an LWO is run on an AS for which the sufficient conditions hold, it may not be the case anymore after the optimization. Furthermore, in [FBR04] we can read that *"First, these sufficient conditions are very strong: they imply that every edge that is on a shortest path to an exit point must have a corresponding iBGP session. Second, the conditions require that redundant route reflectors must be located close to the primary to have a similar view of the best routes, introducing undesirable fate sharing. Finally, we have recently discovered IGP topologies for which this constraint is not satisfiable"*. Another interesting recent work is [BUM08] where the authors propose a new algorithm to design correct iBGP route-reflection topologies. The remaining problem with that method is that it supposes that the set of link weights is known and stable. If the link weights are reoptimized by an LWO for traffic engineering, it is required to re-design the iBGP route-reflector topology from scratch for this new set of link weights, which is really not desirable.

In [VVKB06] they propose a new method to build an iBGP topology made of route reflectors which can guarantee the *complete visibility* property[2]. The *complete visibility* property is defined in [VVKB06] as: *The dissemination of information amongst the routers should be "complete" in the sense that, for every external destination, each router picks the same route that it would have picked had it seen the best route from every other BGP router in the AS.* This technique is a great work and good step toward good route reflector configuration in networks. If that technique is used to design the hierarchy of route reflectors, there is no problem with previously proposed LWOs, as they guarantee that the egress point used by any router is the egress that would be chosen in the iBGP full-mesh case, and this remains true for any set of link weights. But

[1] In that paper they define a correct iBGP configuration to be one that is anomaly-free for every possible set of routes sent by neighboring ASes. They focus on anomalies that can cause the protocol to diverge, and those that can cause path deflections.

[2] The proposed method also guarantees *loop-free forwarding* and *robustness to IGP failures.*

it is not clear whether the technique of [VVKB06] can be used in practice to design all the route reflector configurations. Indeed the algorithm is really not flexible: it proposes one and only one solution which is correct but absolutely not tunable. What happens if the proposed solution still contains too many iBGP sessions? The number of iBGP sessions is reduced by a factor between 2.5 and 5 depending on the network compared to an iBGP full-mesh. But it is not clear that it is sufficient for every network. Another potential problem is the number of route reflectors and the number of levels in the route reflector hierarchy which seems quite high in the generated configurations. This may be considered too complex by some network operators.

In addition iBGP configurations that are currently deployed in the internet contain errors. For example in [FB05] they have analyzed the real-world, deployed BGP configuration of 17 different ASes. They have detected more than 1000 BGP configuration faults that had previously gone undetected by operators.

7.2.2 LWOs

Some methods have been proposed to integrate the BGP hot-potato rule in the optimizer. For example in [ANB05], they recompute the intradomain traffic matrix from the interdomain traffic matrix at each step of the optimizer, choosing for each ingress node the nearest next-hop for each destination prefix. This solution is not correct when the complete visibility property is not respected, as in this case it is not always the nearest next-hop which is used, but the nearest *available* next-hop, which may be very different.

In [BL08] (the proposition of chapter 6) and [Rex06] the proposed method consists in adding some virtual nodes to the intradomain topology to model the reachability of prefixes via multiple egress points. These solutions are based on the fact that the traffic will follow the shortest path based on the link weights from an ingress node to the virtual node modeling the destination prefix, which respect both the IGP based on shortest paths and BGP's hot-potato rule. In this case it is possible to reuse existing Link Weight Optimizers with efficient heuristics on the extended topology. But again this method requires the complete visibility property. If that property is not respected, the chosen next-hop may not be the one which is on the shortest path between the ingress node and the virtual node modeling the destination prefix, as this next-hop may not be available at this ingress node. This means that from an algorithmic point of view it is not sufficient to compute a shortest path to infer the egress node chosen by BGP. A more complex model of the BGP decision process and exchange of BGP messages is required.

7.3 The problem of link weight optimizers and route-reflectors

We will illustrate the problems of link weight optimizers and route-reflectors on the topology of figure 7.1. Suppose that R_1, R_2, R_3 and R_4 are part of the AS we want to engineer, N_1 and N_2 are part of a neighboring AS and can both

reach IP prefix P_1, N_1 has an eBGP session with R_3 and N_2 with R_4. Suppose also that a BGP route advertisement message concerning prefix P_1 is sent on both eBGP sessions ($N_1 \rightarrow R_3$ and $N_2 \rightarrow R_4$) with the same BGP attributes. We will mainly consider the following two configurations of iBGP session:

- There is a full-mesh of iBGP sessions;
- R_2 is the route reflector for R_1, R_3 and R_4.

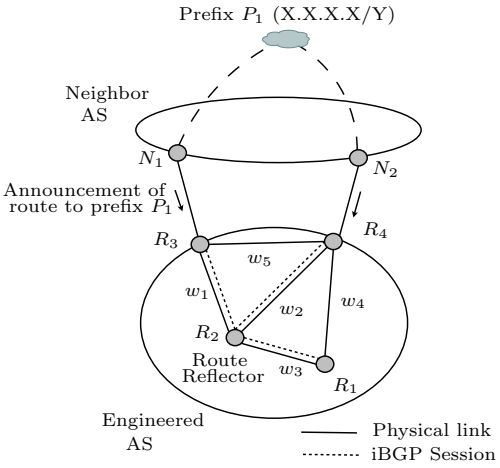

Figure 7.1: Toy Example

First, consider the full-mesh iBGP configuration with the link weight setting of figure 7.2a. R_1, R_2, R_3 and R_4 receive the two available routes on their {e,i}BGP sessions. R_3 chooses the route with N_1 as next-hop while R_4 chooses the route with N_2 as next-hop, both respecting the rule that enforces routers to prefer eBGP-learned routes to iBGP-learned ones. R_1 and R_2 have received these two routes on iBGP sessions, so these routers use the hot-potato rule to choose their best route. R_2 chooses the route with N_1 or N_2 as next-hop depending on the IGP distance between R_2 and respectively R_3 and R_4. For the same reason R_1 chooses between the two available routes depending on the IGP distance from R_1 to R_3 and R_4. For the particular link weights setting of figure 7.2a, R_2 chooses the route with N_1 as next-hop while R_4 chooses the route with N_2 as next-hop. We clearly see that adding a virtual node corresponding to prefix P_1 and two virtual links from N_1 to P_1 and from N_2 to P_1 allows an algorithm to consider that the next-hop chosen by R_2 (resp. R_1) is on the shortest path from R_2 (resp. R_1) to P_1, provided that these two virtual links and the two interdomain links have a weight of 0. This is the idea elaborated in [BL08] (presented in chapter 6).

Now we will analyze the same network where R_2 is the route reflector for R_1, R_3 and R_4. Figures 7.2b, 7.2c and 7.2d do all use these iBGP sessions. The only differences are on the link weights setting. According to the BGP protocol the best routes will be determined as follows. R_3 and R_4 will send their routes to the route reflector R_2 which will run its BGP decision process and thus select one of these depending on the IGP distance from R_2 to R_3 and R_4 respecting

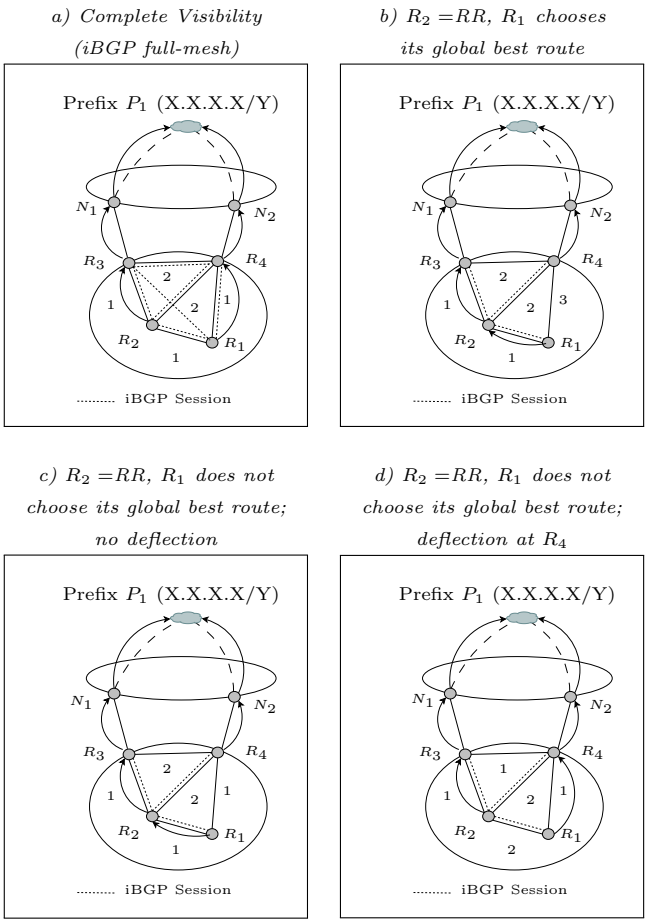

Figure 7.2: Different iBGP configurations and link weights setting

the hot-potato rule. Now R_2 will forward this (and only this) best route to R_1 which will choose the same route as only this one is available at this ingress router. We clearly see that R_1 will use the same egress point as R_2 even though it is not necessarily the nearest egress point for R_1. In that case we say that R_1 has a partial visibility if its *globally* best egress is not visible to it.

Consider the link weights setting of figure 7.2b. In that case R_2 chooses R_3 as egress node, and so does R_1 as only this route is available. In this case, R_1 chooses its *globally* best route as the distance from R_1 to R_3 is 2 while the distance from R_1 to R_4 is 3. This means that with these link weights we have complete visibility as defined in section 7.2 (i.e. each router selects the route which is its *globally* best route).

Now consider the link weights setting of figure 7.2c. In that case R_2 still chooses R_3 as egress node, and so does R_1 as only this route is available, although its *globally* best route is via R_4 as the distance from R_1 to R_3 is 2 while the distance from R_1 to R_4 is 1. Anyway in this case there is no path deflection as the shortest path from R_1 to R_3 is $R_1 \rightarrow R_2 \rightarrow R_3$, which does not cross R_4.

Finally consider the link weights setting of figure 7.2d. In that case R_2 still chooses R_3 as egress node, and so does R_1 as only this route is available. It is still not its *globally* best route as the distance from R_1 to R_3 is 2 while the distance from R_1 to R_4 is 1. But now there is a path deflection. Indeed the shortest path from R_1 to R_3 is $R_1 \rightarrow R_4 \rightarrow R_3$. So R_1 forwards traffic to R_4 thinking that R_4 will forward it to R_3 while R_4 will actually send it directly to N_2. In that case we say that the traffic is deflected by R_4. So the real path of the traffic will be $R_1 \rightarrow R_4 \rightarrow N_2$ and not $R_1 \rightarrow R_4 \rightarrow R_3 \rightarrow N_1$.

Note that the combination of multiple path deflections can lead to forwarding loops inside the AS. We refer to [GW02] for an example of such situation.

The consequences of the presence of route reflectors on the performance of LWOs are the following:

- LWOs that consider complete visibility may lead to non-optimal traffic engineering solutions, as the link utilizations that are predicted by the optimizers do not reflect the ones that will be observed in the network, due to errors in the egress prediction. For example, in case of figure 7.2c, the optimizer will consider the routes of figure 7.2a;
- LWOs that consider complete visibility may introduce path deflections. Indeed one *deflection-free* configuration (e.g. figure 7.2b) can be transformed into a BGP configuration *containing deflection* (e.g. figure 7.2d) simply by changing the set of link weights.

To summarize, we have to consider different cases of route reflector configurations, from the safer to the most dangerous:

1 The configurations with complete visibility for every possible link weights setting. For example configurations generated using the technique described in [VVKB06] or simple iBGP full-mesh configurations.

2 The configurations with complete visibility for the present link weights setting, without guarantee that all possible link weights settings will keep the complete visibility property (e.g. figure 7.2b).

3 The configurations without complete visibility for the present link weights setting, but for which no deflection occurs (e.g. figure 7.2c).

4 The configurations without complete visibility for the present link weights setting, for which simple deflections occur between egresses toward the **same** neighbor AS[3] (e.g. figure 7.2d).

5 The configurations without complete visibility for the present link weights setting, for which simple deflections occur between egresses toward **different** neighbor ASes (e.g. figure 7.2d, but with N_1 and N_2 in two different ASes).

6 The configurations without complete visibility for the present link weights setting, for which multiple deflections occur and form a forwarding loop inside the AS (see [GW02] for such an example).

It can be noted that we make a difference between cases 4 and 5 because simple deflections between two egress routers that connect to the same neighbor AS (and which have the same corresponding ASPATHs) are not potentially the cause of inter-AS loops. Indeed in that case there is no erroneous ASPATH information transmitted to neighboring ASes. We think that this kind of deflection could be allowed by a network operator, as it is not dangerous. Moreover we will see in section 7.6 that allowing this kind of deflection may let the optimizer find a better TE solution in the extended search space.

Case 1 happens when the algorithm of [VVKB06] is used. If run on the topology of figure 7.1, it would probably produce the following iBGP configuration: R_2 and R_4 are both route reflectors for R_1 and R_3. It is not possible to find a topology with only one route reflector that would produce the complete visibility property for every possible link weights setting.

In case 1 we are sure to avoid the problems due to partial visibility and so BGP-aware LWOs (like [ANB05, BL08, Rex06]) can be used. In all the other cases, these optimizers can fail because of potential partial visibility and path deflections. This can be simply explained by the fact that changing the link weights setting can drive from any configuration in the set {2,3,4,5,6} to any other configuration in the set {2,3,4,5,6}. So even if we start a link weight optimization on a configuration of type 2, the resulting link weights can lead to a configuration from type 2 to 6 in the worst case.

The important point is that if the iBGP configuration of a network is not of type 1, it is safer to run a LWO that can deal with partial visibility and avoid (or at least minimize the number of) deflections in the network. The iBGP configuration with optimized link weights setting should ideally be of type {1,2,3,4}, while type {5,6} should be avoided. Indeed type {1,2,3,4} guarantee that *no forwarding loops* will be created.

[3]In fact we should say for which the corresponding ASPATH is the same in both routes, which is generally the case when both routes are received from the same neighbor AS.

7.4 Evaluating the impact of partial visibility on link weight optimizers

In this section we evaluate the impact of partial visibility on the traffic engineering quality of solutions found by LWOs that make a wrong assumption of complete visibility.

To this end we will test a LWO which considers complete visibility on a topology which contains a route reflector giving partial visibility only. We use the LWO presented in chapter 6. We will refer to this LWO as *BGP-CV-LWO* as it correctly takes the BGP hot-potato rule into account when the complete visibility (CV) property is respected. The results should be similar for the tool developed in [ANB05], while we were not able to test it because that tool is not publicly available. In the simulations of this section we use our dataset presented in chapter 3. We have used the technique exposed in chapter 6 to build our model from BGP dumps and netflow data.

7.4.1 Simulation description

The actual iBGP configuration of the network we have studied is an iBGP full-mesh. We have designed an hypothetical simple BGP route reflector hierarchy to simulate what happens with partial visibility. The route reflector hierarchy we consider is the following: (this hierarchy is inspired from [Van05]): one router which has a *central* position in the topology is chosen as the route reflector and all the other routers of the network are clients of this route reflector[4].

For each traffic matrix, we have run *BGP-CV-LWO* which considers hot-potato traffic, but with complete visibility. Values labeled *"predicted"* denote the link loads predicted by this algorithm which assumes optimal hot-potato egress selection. Values labeled *"resulting"* represent the actual link loads resulting from the BGP behavior with possible non-optimal egress selection due partial visibility causing non best route choice for some routers. We have used the C-BGP simulator ([QU05]) to compute the actual outcome of the BGP decision process in this situation.

7.4.2 Simulation results

Figure 7.3 presents the CDFs (cumulative distribution functions) of the maximal link utilization for all the traffic matrices (TMs). We can see that the optimizer predicted that about 75% of the TMs will lead to a maximal utilization below 40% while it is in practice only true for about 45% of the TMs. Note that a maximal link utilization over 100% means that one link is overloaded in the network. The mean error of the optimizer concerning the maximal link utilization is about 7.8%. In the worst case, the maximal utilization is greater than 140% while the optimizer predicted less than 65%. The *BGP-CV-LWO* that was very efficient and precise with complete visibility provides very poor results in this route reflector configuration. If we compare these results with the results of

[4]By choosing the router which has the "central" position we mean that we have chosen the router whose sum of its distance (in terms of number of hops) to all other routers is minimal.

chapter 6 (figure 6.11) we can say that with this route reflector configuration the *BGP-CV-LWO* behaves as badly as a completely BGP-blind LWO. These results show that a more precise model of the BGP decision process is really needed in the link weight optimizer.

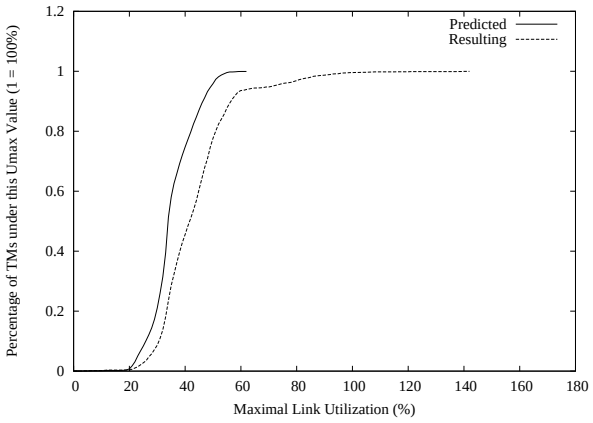

Figure 7.3: CDFs of *"predicted"* versus *"resulting"* U_{max} over all TMs for *BGP-CV-LWO* on the topology which contains a route reflector

Note that we have not just analysed the CDFs of the maximal link utilization. We will summarize our observations concerning the other links. The CDFs concerning the mean link utilization are almost identical, which means that partial visibility has almost no impact concerning the mean link utilization. Concerning the link just after the highest loaded link (which we call the second maximal link utilization), partial visibility has an impact, but which is significantly lower than for the maximal link utilization. Then for the third and the fourth link utilization, the impact is very reduced (almost no impact). This is the reason why we will only consider the maximal link utilization in our subsequent simulations. Because partial visibility has a big impact on maximal link utilization. So that this is the maximal link utilization that we can reduce with a correct model of route reflectors.

7.5 A generic BGP-aware link weight optimizer

We now present *BGP-LWO* a new LWO which can consider any iBGP configuration. As a result it can correctly predict the actual egress node for each prefix. To our knowledge, this is the first link weight computation algorithm that correctly models hot-potato reroutings in the presence of Route-Reflectors (leading to partial visibility) and that detects path deflections.

7.5.1 Description of the algorithm

The algorithm is based on the simulated annealing metaheuristic. It is presented on algorithm 7.1. Here is a brief description of this metaheuristic. Note that T is the current temperature while T_0 is the initial temperature. We start with an

initial solution (line 7 → 10). Then we enter the main loop. At each iteration of the main loop we do the following actions: propose a move, decide to accept the move or not, and record the solution if it is the best solution found so far (lines 14 → 30). If the proposed move does improve the objective function, the move is accepted (lines 18 → 22). If the proposed move does not improve it, the move is accepted with a given probability (lines 24 → 28). As the simulation goes on, the probability of accepting a non-improving move decreases (via the temperature). The higher the temperature T, the higher the probability to accept the move. Every L iterations (one plateau), the temperature is decreased by multiplying it by a cooling factor $\alpha < 1$. At some point, when the number of accepted moves goes under a determined threshold (less than ϵ accepted moves during the last K plateau), the simulation is stopped.

The steps that need to be configured are the following:

- *What is a solution?*

  In our case a solution is a set of link weights (one for each link).

- *How to find/create the initial solution?*

  We decided to start from a random link weights setting.

- *How to find a move?*

  We decided that a move is one random change to one link weight (the link is chosen randomly).

- *Which score to associate with a solution?*

  We use a modified version the $Fortz$ objective function. We just add a penalty for each deflection that happens in the network[5]. This penalty should drive the optimizer to solutions that minimize the number of path deflections. The objective function is now the sum of two separate objectives :

  - The $Fortz$ Traffic Engineering (TE) objective function;
  - The penalty that is directly proportional to the number of path deflections.

  Both functions should be minimized. The value of the penalty associated with each deflection can be tuned to trade-off between both objectives. The network operator should test different values for the penalty and choose one of the resulting solutions. If the network operator absolutely wants to avoid path deflections, the optimizer should return a deflection-free solution if the penalty is set high enough (if such a solution exists, of course). When multiple deflections are combined to form a forwarding loop, the value of the objective function is set to ∞. This will lead the optimizer to discard all these solutions.

Then the value of some parameters of the simulated annealing heuristic (plateau size, initial temperature, cooling factor, stopping conditions) have to

[5]We count one penalty for each deflection from one node to one aggregated prefix. We explain in section 7.5.3 how deflections are detected.

Algorithm 7.1: A Generic BGP-aware LWO based on Simulated Annealing Meta-Heuristic

```
/* x_0, x* and x are the initial,
   the best and the current set of weights */
/* l_0, l* and l are the corresponding
   link loads */
/* F(l) is the objective function */
/* move(x) return a neighbor of x */
x* ← x_0; x ← x_0; T ← T_0;
tm ← computeIntraTrafficMatrix(x);
l ← computeLinkLoads(x, tm);
l^0 ← l; l* ← l;
while not stopCondition do
    nbIter ← 0;
    while nbIter < L do
        x' ← move(x) ;
        tm ← computeIntraTrafficMatrix(x') ;
        l' ← computeLinkLoads(x', tm);
        if F(l') < F(l) then
            x ← x' ;
            l ← l';
            if F(l') < F(l*) then
                x* ← x'; l* ← l';
            end
        else
            ΔF ← F(l') − F(l) ;
            p_k ← e^{−ΔF/T};
            if random() < p_k then
                x ← x'; l ← l';
            end
        end
        nbIter ← nbIter + 1;
    end
    T ← αT;
end
```

be fixed. During our simulations, we have observed that it can be quite difficult to tune all these parameters. But empirical rules (plateau size: same order of magnitude than the number of neighbors, initial temperature: such that about 50-90% of the moves are accepted during the first plateau, cooling factor: around 0.9) are generally a good help and good starting point. Then the parameters have to be tuned to match the particular instance of the problem that we consider (i.e. the topology and the traffic load).

7.5.2 Obtaining the required data

The optimizer requires some BGP and traffic data which is the same as described in section 6.4.4.

7.5.3 Computing the egress node using C-BGP

Our algorithm uses the C-BGP simulator ([QU05]) to compute the egress node for each aggregate of *hot-potato* prefixes. The algorithm gives to C-BGP the topology of figure 7.4 corresponding to the situation of figure 7.1. C-BGP needs as input the physical topology, the link weights and the iBGP configuration. We also configure static routes for interdomain links as this is required by C-BGP. Then the aggregated prefixes are added on the multiple possible next-hop nodes and the C-BGP simulation is run. Finally our algorithm asks C-BGP the Routing Information Base (RIB) of each ingress router concerning each aggregated destination prefix. From this RIB our algorithm can read which route is the best available route. This is the route that would result from applying the specified link weights in the network with this iBGP configuration.

At this step we know the best route from every router inside the AS toward every cluster of hot-potato prefix. Thus we are able to check whether path deflections occur by combining the shortest path information with these best routes. We also check whether forwarding loops have been introduced due to multiple path deflections.

Let us note that the problem is computationally tractable because we do not run the simulator with all the prefixes, but with *aggregated hot-potato prefixes.*

7.6 Evaluation of the proposed optimizer

In this section we will run the new LWO presented in section 7.5 on our dataset[6]. We will refer to our new optimizer which embeds the C-BGP simulator as *BGP-LWO* as it can correctly model the BGP behavior in every iBGP configuration, which is different from *BGP-CV-LWO*, which had been designed with complete visibility in mind.

We first want to evaluate the quality of the solutions found by the new algorithm *BGP-LWO*. To this end in section 7.6.1 we compare it to a state of the art LWO. This comparison is performed in an iBGP full-mesh configuration

[6]The values of the different parameters of the simulated annealing heuristic used in these simulations are the following: $T_0 = 100000$, $L = 50$, $\alpha = 0.9$, $\epsilon = 2$, $K = 3$.

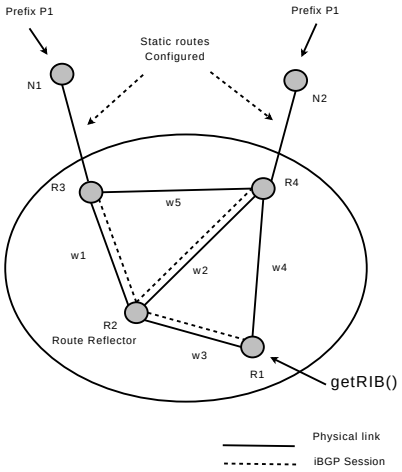

Figure 7.4: Toy Example - in C-BGP

which does not favor one LWO over the other. In section 7.6.2 we compare both algorithms on a configuration with one route reflector. This will allow us to evaluate the TE performance gain of *BGP-LWO* over *BGP-CV-LWO* which wrongly assumes complete visibility. Note that for a fair comparison we have not included a path deflection penalty in *BGP-LWO* in the simulations of sections 7.6.1 and 7.6.2 as of course *BGP-CV-LWO* does not consider path deflections either. Then in section 7.6.3 we include a penalty for path deflections in the objective function. These simulations test the ability of *BGP-LWO* to avoid path deflections. Finally section 7.6.4 evaluates the TE performance gain that can be realized when the network operator allows *non-dangerous* path deflections.

7.6.1 Quality of the new optimizer

First we would like to evaluate the quality of the solutions found by our new LWO. This point is examined by running our new LWO (*BGP-LWO*) and *BGP-CV-LWO* on the whole dataset considering an iBGP full-mesh. *BGP-CV-LWO* is based on the heuristics of [FT04] which we consider as the state of the art. Both algorithms should provide good results on this dataset with this iBGP configuration. Figure 7.5 presents this comparison. Values are sorted according to *BGP-CV-LWO*. Note that values of maximal link utilization under 33.3% are not very important in this comparison as the objective function used is linear under 33.3%. This means that there is no significant penalty associated with links used less than this value and so the optimizer will not really try to reduce a maximal link utilization under this value. We can see on the figure that both optimizers provide solutions of similar quality. This means that the solutions found by *BGP-LWO* are quite good, at least on that dataset.

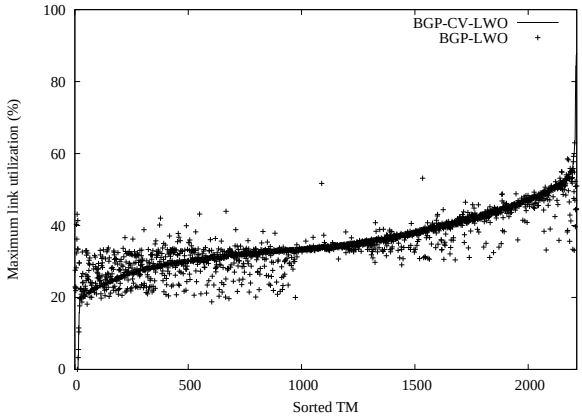

Figure 7.5: Evaluation of the quality of the solutions found by *BGP-LWO* compared to *BGP-CV-LWO* in an iBGP full-mesh configuration

7.6.2 Actual prediction of the egress point

Now we evaluate the quality of our new LWO (*BGP-LWO*) for what it has been designed for, which is when the iBGP topology is not a full-mesh. In that situation it should perform better than other LWOs which have not been designed to work in these situations. We have run our *BGP-LWO* on the full dataset supposing that there is a route reflector in the AS (the iBGP configuration presented in section 7.4). We compare these results with the results of *BGP-CV-LWO* (the optimizer of chapter 6). Figure 7.6 presents the CDF of maximal link utilizations. We can see that *BGP-LWO* solves the problems presented in section 7.4 and performs quite well. This means that it correctly predicts the egress points that will be chosen by the routers. This is quite logical as it was designed for, but these simulations confirm our expectations.

The mean reduction of maximal link utilization obtained by *BGP-LWO* is about 7.6% but can be as high as reducing the maximal link utilization from more than 140% to about 90%.

7.6.3 Minimizing the number of path deflections

As explained in section 7.5, *BGP-LWO* detects and forbid solutions that lead to intra-AS forwarding loops due to the combination of multiple path deflections. A network operator should also avoid (or at least minimize) simple deflections[7] inside its network as these could lead to inter-AS forwarding loops in the worst case.

To this end the methodology that we have adopted is the following. The optimizer is allowed to consider solutions with simple deflection during the execution of the algorithm. So it is easy to find an initial solution and to propose a move. But we have included in the objective function a penalty for each deflec-

[7]Here we define simple deflections as deflections that are not part of an Intra-AS forwarding loop.

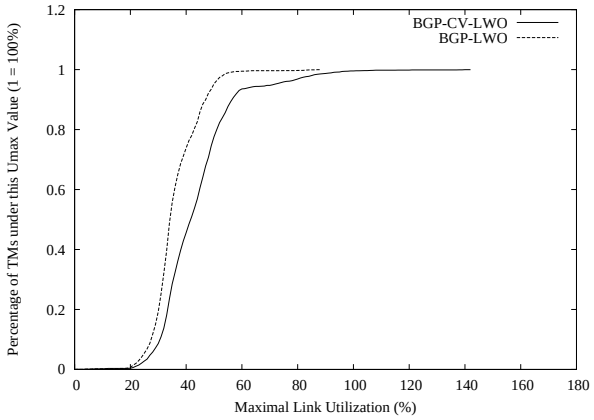

Figure 7.6: CDFs of U_{max} over all TMs for *BGP-CV-LWO* and the *BGP-LWO* on the topology which contains a route reflector

tion. So during the execution of the algorithm the number of path deflections should decrease. If the penalty for each deflection is high enough the optimizer will first try to minimize the number of path deflections. If the number of path deflections is 0 at the end of such a simulation we know that a solution without path deflection exists. If it is not the case, we know the lower bound concerning the number of path deflections[8]. And if the solution which completely avoids path deflections is too bad concerning TE objectives, the network operator can choose (at his/her own risks) to allow some path deflections to improve the solution with respect to TE objectives, by tuning the value of the path deflection penalty.

We will see that on this dataset we are able to completely avoid path deflections while keeping good TE performance.

An execution of *BGP-LWO* without penalty associated with path deflections is shown on the left column of figure 7.7. These graphs present the evolution of different components of the objective function during the execution of the algorithm (from the first iteration -the leftmost point- to the last one -the rightmost point-) for the best solution found so far. We see that the simulation stops after 2500 iterations. The first graph shows the value of the TE part of the objective function, which is supposed to reflect the quality of the load balance (the lower the value the lower the link utilizations). This is quite abstract as this value has no direct physical meaning. This is why we also present the maximal link utilization (the second graph), while this is not directly minimized by the optimizer. This value gives a physical idea of the TE quality of the solution. The third and last graph presents the number of path deflections. We see that the algorithm stops on a solution which induces 43 path deflections. We can conclude from this simulation that the best solution for TE induces a quite high number of path deflections.

[8]This is not absolutely true as the simulated annealing heuristic does not guarantee to find the **best** solution but only a **good** solution. So it may be possible that this execution gives a value which is not the **global** lower bound.

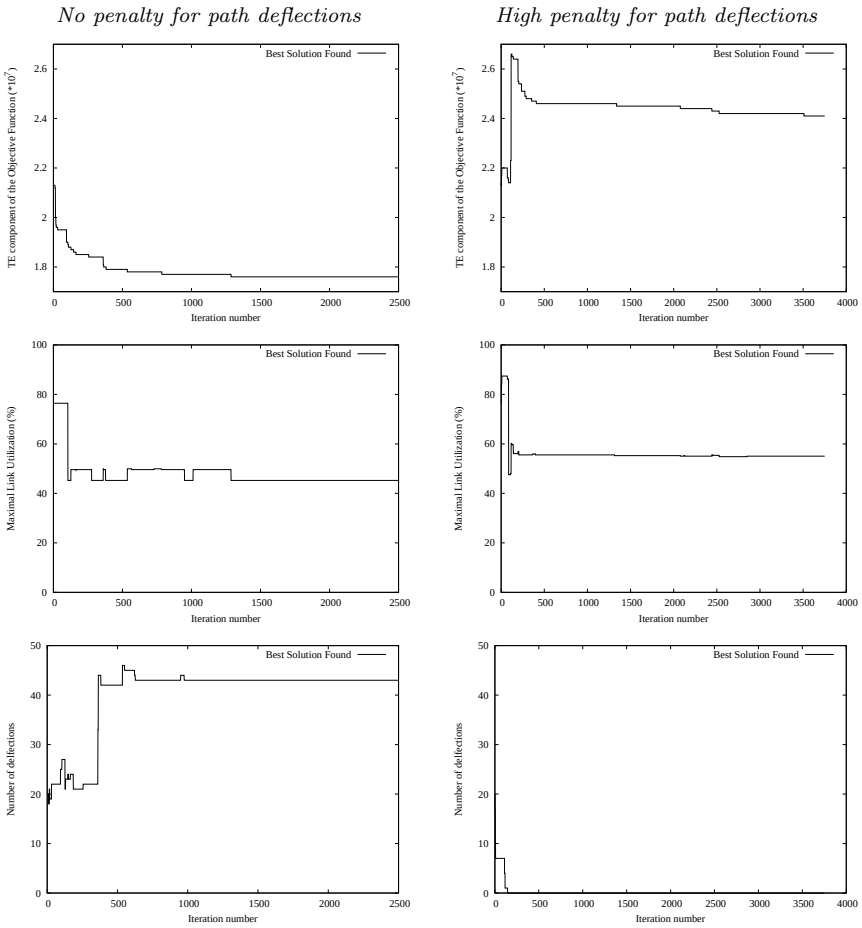

Figure 7.7: Evolution of the Best Solution during the execution of the algorithm for one traffic matrix (part 1)

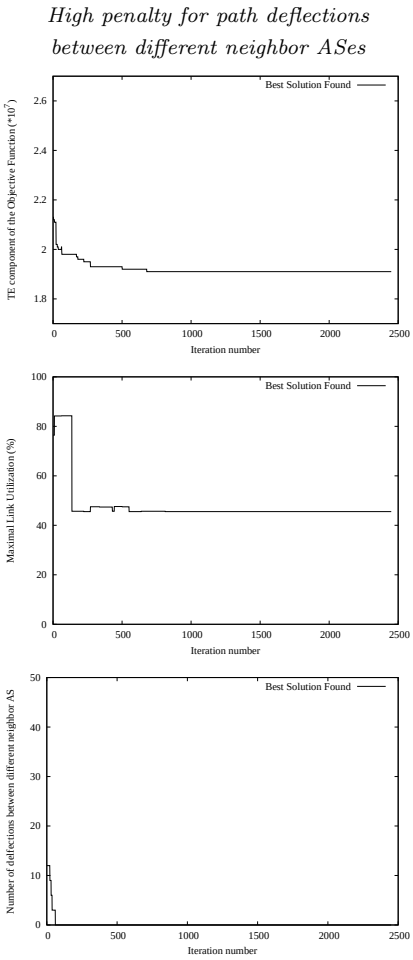

Figure 7.8: Evolution of the Best Solution during the execution of the algorithm for one traffic matrix (part 2)

	TE Objective Function ($*10^7$)	Maximal Utilization	Nb Deflections
No penalty	1.76	45.24%	43
	1.91	45.53%	9
	1.98	45.54%	3
High penalty	2.41	55.05%	0

Table 7.1: Trade-off between load balance and number of deflections

We have also run *BGP-LWO* on the same traffic matrix while giving a very high penalty to each path deflection. The results of this execution should let us know if it is possible to find a solution (i.e. a set of link weights) inducing no deflection at all. Indeed the number of path deflections at the end of this simulation will give the lower bound concerning path deflections. The execution of the algorithm with these parameters is presented on the right column of figure 7.7. The good news is that the algorithm rapidly finds a solution with no deflection (after about 200 iterations). During these first 200 iterations, the TE component of the objective function was not considered by the optimizer (second order of magnitude in the objective function) and the value of this component has been increased from about $2.2 * 10^7$ to $2.65 * 10^7$. After the 200^{th} iteration, as it is not possible anymore to decrease the number of path deflection (0 is obviously a lowest bound), the algorithm tries to minimize the TE component of the objective function while keeping the number of deflections to 0, which is less easy than when there were no constraint on the number of deflections. As a result, at the end of the simulation, the number of deflections for the best solution found is 0, but the corresponding values of TE component and maximal link utilization are quite significantly higher than in the case with no penalty for path deflections. This is quite logical as requiring no deflection limits the search space of the optimizer. It has to find the best TE solution from the set of solutions inducing no deflection while the preceding execution of the algorithm could find the best TE solution from the complete set of solutions, no matter how many deflections would be induced.

Varying the value of the penalty associated with each deflection provides us trade-off solutions between a good TE state and a minimum number of path deflections. These solutions are found in table 7.1. The first line (No penalty) is the solution found at the end of the execution of *BGP-LWO* for the left column of figure 7.7 while the last line (High penalty) is the solution found at the end of the execution of the *BGP-LWO* for the right column of figure 7.7. We see that allowing a low number of deflections (3) allows the optimizer to improve the TE quality of the solution (from $2.41 * 10^7$ to $1.98 * 10^7$ for the TE component of the objective function or from 55.05% to 45.54% for the maximal link utilization).

7.6.4 Allowing deflections between two routes having the same ASPATH

As we have seen in section 7.3, simple deflections between egress routers toward the same neighbor AS (if the two corresponding routes have the same ASPATH) is not a problem as this cannot potentially lead to forwarding loops. So we could

take advantage of this point and avoid only path deflections between different ASes. This will put less constraints on the optimizer, which could allow it to find a better solution for TE. This is confirmed by the execution of the algorithm which is presented on figure 7.8. We can see that the optimizer finds in less iterations (2450 instead of 3750) a better solution ($1.91 * 10^7$ instead of $2.41 * 10^7$ for the TE component of the objective function and 45.52% instead of 55.05% for the maximal link utilization) while inducing no *dangerous* path deflections. This means that in some cases it may be interesting to be flexible with *non-dangerous* path deflections. Avoiding all path deflections may be too restrictive.

7.7 Conclusion

Link Weight Optimizers try and minimize a traffic engineering objective function based on link utilizations. Therefore, for a precise optimization they need accurate estimations of all link utilizations resulting from the application of the optimized weights in the AS.

The first generation of LWOs was imprecise because these algorithms were based on the assumption that the intradomain traffic matrix was invariant when link weights were changed. This optimization thus neglected the effect of BGP's hot-potato rule, which may modify the intradomain traffic matrix, resulting in wrong estimations of actual link utilizations.

The second generation of LWOs did take BGP's hot-potato rule into account, but did not consider cases where routers have only partial BGP visibility due to route reflectors. The consequence is that these LWOs may wrongly predict the egress node for some traffic, and they do not take path deflection into account. The consequence is again an incorrect estimation of link utilizations.

Considering that complete visibility cannot always be affordable in every AS, we have studied in detail the impact of partial visibility on LWOs. We have shown that if route reflectors are present in the AS and lead to partial visibility, previously proposed BGP-aware LWO methods may behave poorly. Furthermore, these LWOs may also introduce path deflections, which in turn may lead to forwarding loops.

The main contribution of this chapter is a new LWO algorithm, embedding the C-BGP simulator in its routing model, that always computes the correct link utilizations in any possible iBGP configuration. An additional asset of our proposed LWO is its ability to avoid path deflections when possible, or otherwise to minimize their number. In any case our LWO always avoids the creation of intra-AS forwarding loops due to multiple path deflections. The efficiency of our LWO with respect to other LWOs has been assessed on a real dataset from an operational network.

8

Can Forwarding Loops Appear when Activating iBGP Multipath Load Sharing ?

In this chapter we analyze the possible consequences of activating iBGP multipath load sharing in a given domain (or AS), which can be valuable for BGP-aware LWOs. Indeed iBGP multipath load sharing allows for load balancing over multiple exit routers. It has been stated that interdomain routing loops may appear when activating this feature. We show that under reasonable assumptions (which reflect commercial relationships between ASes) such routing loops cannot appear. Furthermore we show that even if theses assumptions are not met, routing loops can only be transient.

8.1 Introduction

Typically link weight optimizers use ECMP (Equal Cost Multi-Path) to split the traffic on multiple paths between one ingress node and one egress node. Using ECMP has multiple advantages. For example ECMP can be used to improve IP restoration ([ICBD04]). It is also a flexible routing technique as with ECMP it is possible to split a big flow on multiple paths instead of routing it entirely on a unique path. Usually ECMP allows a good engineering of the network.

While it is considered valuable to split traffic on multiple paths inside a domain, splitting traffic on multiple interdomain paths is rarely envisaged. However this would make the Internet more efficient and robust, improving security, reacting to failures, and balancing load ([HR08]).

In practice BGP typically chooses one (and only one) path among its multiple available ones. Although in an AS some destination prefixes are reachable via only one egress point, it is frequent that most of the prefixes (typically provider prefixes) are reachable via multiple BGP-equivalent routes (for example if the AS has multiple links connecting its providers). Using classical BGP one of these routes is chosen via the Hot-Potato criterion or a tie-break at a later stage of the BGP decision process. But it is also possible to configure BGP to allow the network operator to split traffic amongst multiple BGP-equivalent routes

(as we have seen in chapter 6).

But the situation is not as beautiful as it seems. Indeed splitting traffic amongst multiple available BGP-equivalent routes which may have a different AS-level paths can cause problems as explained in [HBJ06]. In that paper the authors state that forwarding loops could appear and they propose a solution. In this chapter we show that contrary to what may be expected and under reasonable assumptions, forwarding loops should not appear in any case. These assumptions are based on the BGP router configurations that typically reflect commercial relationships.

The chapter is organized as follows. In section 8.2 we briefly describe why forwarding loops could appear with iBGP multipath load sharing. Section 8.3 presents the BGP configuration we assume in this chapter. These are natural BGP configurations that should be respected in all ASes. We show in section 8.4 that if these assumptions hold, no forwarding loops can appear. In sections 8.5 and 8.6 we analyze what happens if the assumptions we made about BGP are not respected. Indeed even if they should be respected in all ASes it is impossible to be sure of that. We show in section 8.5 that even in this case no forwarding loops can appear when activating iBGP multipath load sharing. These can only appear at a later stage if the BGP configuration of an AS is changed. We show in section 8.6 that even in this case forwarding loops are only transient. Section 8.7 concludes the work.

8.2 Forwarding Loops?

iBGP multipath load sharing has been presented in chapter 6. We will not present it again here, but we want to recall that iBGP multipath load sharing is the ability to balance the load on multiple egress points if the corresponding multiple BGP routes are equivalent and that the egress points are at the same IGP distance.

We have also to note that BGP ([Ste99]) includes the following loop prevention mechanism. When an AS receives a route whose ASPATH contains its AS number, it discards the route. This assumes that the ASPATH contains a full list of all the ASes along the path used to forward traffic toward this destination. If part of the ASPATH information is lost, this mechanism does not work anymore.

In [HBJ06] we can read that *Most of the current BGP implementations upon receiving multiple equal cost BGP routes from different peers can insert all of them (or a subset depending upon the local policies) in their forwarding table. This can be done to locally split the traffic across several paths. However, because BGP in its current state can only advertise one path to its peers, an implementation MUST choose from one of the best paths that it is using for the advertisement. This has implications for the BGP peers that receive such advertisements from ECMP capable BGP speakers. In the worst case it can lead to potential loops if the entire path information is not advertised to the peers.*

In [HBJ06] the authors present a first method to avoid forwarding loops using BGP AS_SET and AS_SEQUENCE. In next sections we analyze what happens if this method is not used and only one ASPATH is announced to other ASes.

Contrary to what may be expected, we show that forwarding loops should not appear when using iBGP multipath on different ASPATH routes.

Definition 1. *A packet is trapped in a forwarding loop if there is a cycle of routers such that each router on the cycle forwards the packet to the next router on the cycle, leading the packet to be infinitely forwarded on the cycle.*

Of course forwarding loops should be avoided in practice. Note also that in IP networks, the time to live (TTL) field of the IP header will force routers to drop a packet which is trapped in a forwarding loop.

Definition 2. *A provider loop (for a particular destination prefix) is a cycle of ASes such that each AS on the cycle is the provider of the next AS.*

Note that a provider loop is also a customer loop if the cycle is analysed in the opposite direction.

8.3 BGP Model Used

In this chapter we consider the following common BGP configurations.

Assumption 1. *We consider import/export rules which state that ([Sob05], [GR01], [Ala96]) :*

- *an AS does not export to a provider or peer routes that it learnt from other providers and other peers;*
- *an AS can export to its customers any routes it knows of.*

This assumption (1) reflects that an AS does not want to provide transit services between its providers and peers.

Assumption 2. *We consider that routes learnt from customers should be preferred to routes learnt from either providers or peers, leaving ASes latitude to assign relative preferences among customer routes, and among peer and provider routes.*

This assumption (2) is the preference rule suggested in Guideline A of Gao and Rexford [GR01]. This is a logical assumption for commercial relationships. Indeed an AS earns money for the traffic it sends on its customer links while it does not earn money for the traffic it sends on its peer links and it pays for the traffic it sends on its provider links. So it should always prefer to send traffic to its customers than to its providers or peers when it has the choice.

Our last assumption is the following (this is also assumed in [GR01]).

Assumption 3. *We assume that there is a hierarchical customer-provider relationship among ASes.*

This is equivalent to saying that there is no provider loop in the AS-level topology.

8.4 When Do Routers Use the BGP Loop Prevention Mechanism?

The BGP loop prevention mechanism implemented in a BGP router consists of discarding routes whose ASPATH contains the AS number of the router[1] ([Ste99]). When and how does this situation happen?

For this situation to happen, we have to be in the case of figure 8.1. AS_X receives a route for a destination prefix from AS_1. It announces this route to AS_2. Later AS_X receives back this route from AS_3 and discards the route because its AS number appears in the ASPATH.

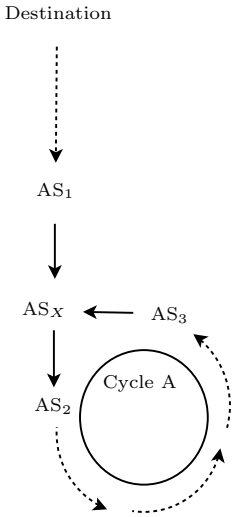

Figure 8.1: AS topology

We will demonstrate that this situation never happens if Assumptions 1, 2 and 3 are respected. We divide the problem into different cases, depending on the commercial relationship between AS_X and its neighbouring ASes for the particular destination prefix we consider. Note that applying this reasoning to each prefix known by AS_X allows us to generalize our result.

8.4.1 AS_1 is a provider or a peer of AS_X

AS_X has received the route from a provider or peer. So AS_X will export this route to AS_2 only if AS_2 is one of its customers (applying Assumption 1). Following the same reasoning the route is announced from AS_2 hop by hop to AS_3 and finally back to AS_X if all these links are provider to customer links. If it is not the case the route is stopped before coming back to AS_X. So AS_3 is a provider of AS_X and cycle A is a provider loop. This situation should not happen as we assumed in section 8.3 that there is a hierarchical customer-provider relationship among ASes (Assumption 3).

[1]Note that this loop detection can also be performed on the sender-side. In this case a BGP router will not announce a route to a neighboring router if its AS number is in the ASPATH of this route.

Now if cycle A is a provider loop (meaning that Assumption 3 is not respected), a forwarding loop could appear if AS_3 is preferred to AS_1 which are both providers. In this case BGP loop prevention mechanism will discard the route from AS_3 which could be chosen if this mechanism were not present.

8.4.2 AS_1 is a customer of AS_X

As AS_1 is a customer, AS_X can announce the route on all its BGP sessions (Assumption 1). So AS_2 can be a customer, a peer or a provider of AS_X. We consider all these cases.

8.4.2.1 AS_2 is a customer of AS_X

In this case, following the same kind of reasoning as in section 8.4.1, the route will come back to AS_X only if all the links from AS_2 to AS_3 and back to AS_X are provider to customer links. So this implies that cycle A is a provider loop (meaning that Assumption 3 is not respected).

The situation is a little bit different than in section 8.4.1, because anyway, if this situation happens, AS_X will always prefer the route from AS_1 which is a customer when compared to the route from AS_3 which is a provider (Assumption 2). In this case Assumptions 1 and 2 are sufficient to guarantee the absence of forwarding loops.

8.4.2.2 AS_2 is a provider of AS_X

In this case AS_2 has received the route from AS_X which is one of its customers and so it can announce it on all its BGP sessions (Assumption 1). Thus AS_3 can be a customer, a peer or a provider of AS_X. We consider all these cases.

a) AS_3 is a provider or a peer of AS_X In this case, AS_X will prefer the route coming from AS_1 (which is one of its customer) to the new route coming from AS_3 (which is a provider or a peer) (Assumption 2).

In this case Assumptions 1 and 2 are sufficient to guarantee the absence of forwarding loops. Note also that this (non-problematic) situation may happen without provider loop.

b) AS_3 is a customer of AS_X For AS_3 to announce the route to AS_X (which is its provider), it must have received this route from one of its customers (applying Assumption 1). By extending this reasoning we can deduce that the route has been propagated hop-by-hop on customer to provider links from AS_2 to AS_3. Otherwise the route would have been stopped between AS_2 and AS_3. In this case cycle A is also a provider loop and this should not happen (Assumption 3).

Note that if this situation happens (meaning that Assumption 3 is not respected), a forwarding loop could appear if AS_3 is preferred to AS_1 which are

Line	AS_1	AS_2	AS_3	Provider loop	Potential Forwarding loop
1	Provider	Customer	Provider	YES	No if BGP prevention
2	Peer	Customer	Provider	YES	NO[2]
3	Customer	Customer	Provider	YES	NO
4		Provider	Provider or Peer	NO	NO
5			Customer	YES	No if BGP prevention
6		Peer	Provider	NO	NO

Table 8.1: All possible configurations (referring to fig. 8.1) leading AS_X to receive a route advertisement whose ASPATH contains its own AS number.

both customers (which respect Assumption 2). In this case BGP loop prevention mechanism will discard the route which could be chosen if this mechanism were not present.

8.4.2.3 AS_2 is a peer of AS_X

In this case AS_2 will announce this route only to its customers (Assumption 1). So the route will be announced hop-by-hop on provider to customer links to AS_3 and then to AS_X (Assumption 1). AS_3 is a provider of AS_X and thus AS_X will prefer the route from AS_1 which is one of its customer to the route from AS_3 which is one of its provider (Assumption 2).

The conclusion is the same as in preceding paragraph labeled a).

8.4.3 Summary

Table 8.1 presents all the possible router configurations that result in AS_X receiving a route whose ASPATH contains its AS number. In all other router configurations it is not possible for AS_X to receive such a route.

Note that only two of these configurations could result in forwarding loops if BGP prevention mechanisms were not enabled. These two configurations are the lines marked with the label "No if BGP prevention" in the "Potential Forwarding loop" column (lines 1 and 5). Note that these two configurations imply that a provider loop is present in the network, which was supposed not to happen as stated in Assumption 3. Thus we can say that the BGP loop prevention mechanism is a kind of watchdog avoiding forwarding loops in misconfigured networks (i.e. networks which do not respect our Assumptions).

Anyway we cannot be 100 % sure that our assumptions are respected in the whole Internet. This is why the BGP loop detection check is still useful in today networks. In the next sections, we will analyze what happens if our assumptions are not respected and what is the impact of this point on the activation of iBGP multipath load sharing.

[2]This is due to the fact that usually routes received from peers are preferred to routes received from providers even if this is not included in our assumptions. If we do not assume this preference rule, line 2 should just be merged with line 1.

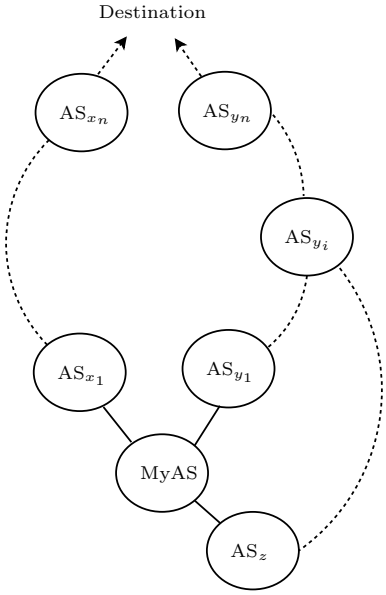

Figure 8.2: iBGP mutipath AS topology

8.5 No Forwarding Loop When Activating iBGP Multipath Load Sharing

In this section we would like to analyse whether activating iBGP multipath load sharing can result in a forwarding loop or not. Indeed a BGP router which activates iBGP multipath on multiple routes will announce only one of these routes to its neighboring ASes. If later on, one AS on one route that has not been announced receives back this route, its BGP loop detection mechanism will be unable to detect the loop[3]. For such a situation to appear we have to be in the case of figure 8.1 in which one of the routers between AS_2 and AS_3 on cycle A enables iBGP multipath on at least two routes, one going to the destination via AS_X and another route in which AS_X is not present. Such a general topology is depicted on figure 8.2, where AS_{y_i} is the AS_X of figure 8.1, $AS_{y_{(i+1)}}$ is AS_1, cycle A is AS_{y_i} ... AS_{y_1} MyAS AS_z ... AS_{y_i} and MyAS is the AS on cycle A which enables iBGP multipath load sharing on multiple available routes : AS_{y_1} ... AS_{y_n} and AS_{x_1} ... AS_{x_n} which does not contain AS_{y_i}. We will show that even with such a topology no permanent forwarding loop can be installed. As this topology is built to reflect all the possible topologies that can lead to a permanent forwarding loop, this will imply that no forwarding loops can be created when using iBGP multipath load sharing. Note that optionally AS_{y_i} could be merged with AS_{y_1} and/or AS_z. Our reasoning can also be applied if iBGP multipath load sharing is used on more than one additional path to the destination in which AS_X is not present.

Suppose that at time $t = t_0$ iBGP multipath load sharing is not activated

[3]Of course this can only happen if at least one of our assumptions is not respected, as it has been shown in section 8.4.

in the network and that MyAS has two BGP-equivalent routes w.r.t. criteria 1 to 6 whose ASPATH are AS_{x_1} ... AS_{x_n} and AS_{y_1} ... AS_{y_n}. One of the two available routes is chosen with some tie-break and this route is announced to AS_z. Suppose now that at time $t_1 > t_0$ we do activate iBGP multipath load sharing on these two routes and that we continue to announce the same route to AS_z. We will show that in this case no forwarding loop is created at time t_1. Indeed the route that was announced at time t_0 was either the route received from AS_{y_1} or the route received from AS_{x_1}. If it was the route received from AS_{y_1} no forwarding loop can be created because AS_{y_i} will see its AS number in the ASPATH received from AS_z. If it was the route received from AS_{x_1}, a forwarding loop cannot be created at time t_1. Indeed the route announced to AS_z is the same at time t_1 than at time t_0. So if AS_{y_i} prefers the route coming back from MyAS via AS_z to the route received from $AS_{y_{(i+1)}}$, it would already have chosen this route at time t_0 and the route with ASPATH AS_{y_1} ... AS_{y_n} would not have been available at MyAS.

8.6 Forwarding Loops Can Only Be Transient

Now suppose that in the preceding example, at time $t_2 > t_1$, the route selected by BGP at router AS_{y_i} changes. There are two possibilities. Either both routes are used by activating iBGP multipath load sharing at router AS_{y_i} or the route selected by BGP is now the route received back from MyAS via AS_z instead of the route received from $AS_{y_{(i+1)}}$. We will analyse both cases separately.

8.6.1 Both routes are selected and used

We will see that this situation is impossible. Indeed this implies that at time t_2, AS_{y_i} activates iBGP multipath load sharing and splits its traffic on its two available routes (the route received back from MyAS via AS_z and the route received from $AS_{y_{(i+1)}}$). But iBGP multipath load sharing cannot select these two available routes as these do not have the same ASPATH length ($|AS_{y_i}\ ...\ AS_{y_n}| \leq |AS_{y_1}\ ...\ AS_{y_n}| = |AS_{x_1}\ ...\ AS_{x_n}| < |AS_{y_i}\ ...\ AS_z\ MyAS\ AS_{x_1}\ ...\ AS_{x_n}|$[4]). Indeed one condition for iBGP multipath load sharing to be activated on multiple routes is that these routes are equivalent w.r.t. BGP criteria 1 to 6, which implies equality of ASPATH lengths (via criterion 2).

8.6.2 The route received back from MyAS via AS_z is now the best route

AS_{y_i} has to change its BGP policies (i.e. its local pref values) for BGP to select the route received back from MyAS via AS_z as best route instead of the route received from $AS_{y_{(i+1)}}$. Indeed the local prefs are the only way to force BGP to select a route whose ASPATH is longer (see BGP decision process). In this case a forwarding loop is created. But as AS_{y_i} now has changed its route, it must withdraw the old route and advertise the new one to $AS_{y_{(i-1)}}$ and so hop by hop to MyAS. When MyAS receives the new route, it can detect the loop because its

[4] $|ASPath|$ denotes the number of ASes of $ASPath$.

AS number appears in the ASPATH. So MyAS will stop using the route received from AS_{y_1} and the forwarding loop is stopped. Note that at this time the router of MyAS which detects and stops the forwarding loop should alert the network operator that at least one of our assumptions is not respected somewhere. With such an alert the network operator could analyze the situation and look for the cause of the problem. Indeed this means that one of our 3 assumptions is not respected.

8.7 Conclusion

We have analyzed how forwarding loops can appear in current BGP networks. We have shown that forwarding loops should not appear even if part of the ASPATH information is discarded, which can be the case when using iBGP multipath load sharing for routes with different ASPATH. Indeed we have shown that BGP configurations reflecting commercial relationships ensure that no forwarding loops will appear. Anyway as it is not possible for a network operator to verify the good configuration of all the involved ASes, we have analyzed what would happen in this case (i.e. if BGP configuration would not reflect commercial relationships). We have shown that even in this case, a forwarding loop cannot appear immediately after activating iBGP multipath load sharing. The forwarding loop could only appear if in addition to the aforementioned conditions, some ASes change their policies in a particular way. Moreover we have shown that even in this case, if a forwarding loop appears, it is only transient.

This leads us to conclude that activating iBGP multipath load sharing for routes with different ASPATH is not as dangerous as it may seem at first glance.

9

MPLS Traffic Protection

We now consider a problem which is quite different from the preceding chapters. This chapter focuses on the protection of virtual circuits (Label Switched Paths, LSPs) in a (G)MPLS (Generalized Multi-Protocol Label Switching) network. The proposed algorithm is designed to protect traffic with strong delay requirements such as EF (Expedited Forwarding) ordered aggregates in a Diff-Serv domain. Indeed, for this type of application, we need fast restoration in case of failure. The duplication of all the packets in a 1+1 end-to-end restoration scheme consumes a large amount of bandwidth. Furthermore, end-to-end recovery with bandwidth sharing schemes are usually considered to be far too slow. Local fast-rerouting is a solution which can compete with restoration times and bandwidth consumption offered by SONET self-healing rings. Our scheme includes a sophisticated resource sharing mechanism based on the concepts of "backup-backup sharing" and "backup-primary sharing". The path selection algorithm is also designed to efficiently reduce the resource usage. Moreover, when considering LSPs at different preemption levels, our algorithm is able to correctly calculate the amount of bandwidth that can be preempted despite the sharing of resource. We show that our approach, though local, can compete with the state-of-the-art end-to-end recovery schemes in terms of resource consumption. The major contribution of our scheme, the "backup-primary sharing", was then also used in the context of end-to-end recovery and improved its performance substantially. To be able to save a maximum amount of bandwidth in a decentralized implementation, the nodes that compute backup LSPs need to obtain a certain amount of link-state information. We propose a solution where the nodes learn almost all the information they need with RSVP messages. This drastically reduces the information that needs to be flooded in the whole network and is the first scalable decentralized solution capable of sharing a large amount of bandwidth.

9.1 Introduction

Today, link and node failures are frequent [MIB+04]. When an element of the network fails, all the traffic passing through this element is lost (at least during the recovery procedure), which can really decrease the Quality of Service (QoS) perceived by all the users of the network. For this reason, there is a real need to develop algorithms and protocols that will allow the network to quickly recover from any failure it may encounter. Such algorithms were studied since virtual circuits exist, but they were mainly solved off-line with the use of optimization theory (Linear Programming/Integer Programming problems). For distributed on-line traffic engineering, the network must be able to compute and establish a path from an ingress router to an egress one and to protect it against failures, based on a request defining the bandwidth and QoS requirement of the traffic.

9.1.1 Related works

A classical way to achieve reliability is to use schemes referred to as 1+1 and 1:1 protection (in [SMH03], V. Sharma et al. introduce these concepts in the context of MPLS). For each LSP we want to establish, we compute two completely disjoint paths from the ingress to the egress. The best of the two is the primary path, the other is the backup path. In the 1+1 scheme both paths are used simultaneously: all packets are duplicated at the ingress and sent on both paths. The egress node continuously monitors both inputs and selects the "best" one. This way of ensuring protection has the advantage of fast receiver-driven recovery upon failure but is of course very costly in terms of bandwidth. An approach to limit the cost of the 1+1 protection is presented in [KKL02]. Since sharing cannot be used, the goal is to achieve efficiency by improving path selection. In order to fulfill this objective, the authors transpose the minimum interference criteria (used in MIRA [KL00]) by replacing the concept of maximum flow by maximum 2-route flow. Following this criteria, they try to route demands such that the reduction of the maximum 2-route flow for the different ingress-egress pairs is as small as possible. Doing so, they hope that it will be possible to accept more demands in the future. This is confirmed by the simulations that show a low rejection ratio for this solution.

In the 1:1 scheme, only the primary path is used to forward packets while the backup path is in "standby" mode. If a failure occurs (on the primary path), a message is sent to the ingress which swaps the backup and the primary path. Obviously the 1:1 protection induces far more delay than the 1+1. Failure has to be detected, a message must propagate to the ingress node which must then swap active and standby paths. For this reason other approaches have also been envisaged. In fact restoration strategies can be divided into two classes: end-to-end recovery and local recovery (often called "fast re-routing"). In a local scheme re-routing is handled by the node directly preceding the failure on the primary path or more generally by a node "close" to the failure. The idea is to establish a set of backup LSPs each one protecting the primary path against the failure of one particular node (or link). Experimental results demonstrate that GMPLS with RSVP can be applied to optical/electrical mesh networks to yield ultra-fast provisioning and restoration times competitive with SONET rings [LYDW01].

The advantage of the 1:1 solution is that significant bandwidth saving can be realized. Indeed if we assume that only a single failure may happen in the network at any given time, not all backup paths can be activated simultaneously. Resources that must be reserved for independent backup paths can thus be shared. Some information must be made available to the device that computes primary and backup paths if we want to achieve the best possible sharing of resources [KL01, LWKD02, QX02]. The authors of [LWKD02] present an algorithm able to protect the network against link failures with a relatively small amount of data. The extra bandwidth consumption is limited to 63-68%, which is a significant improvement compared to previous methods. M. Kodialam and T.V. Lakshman show in [KL01] that a partial information scenario which uses only aggregated and not per-path information can achieve efficient dynamic routing of locally restorable bandwidth guaranteed path. The partial information flooded in the whole network specifies the part of each reserved link bandwidth which is reserved by primary paths and the part reserved by backup paths. The same kind of reasoning as [KL01] in an end-to-end instead of a local protection scheme is applied in [KL03]. Our work is related to [KL01] and presents some similarities. One of the main advantages of our solution compared to the solution in [KL01] is that we can achieve the efficiency of the complete information scheme with nearly the cost of the partial information scheme.

Paper [BBO+03] integrates QoS constraints in the computation of backup paths. Y. Bejerano et al. consider both bottleneck QoS constraints such as bandwidth, and additive QoS constraints such as delay and jitter. In that paper, the authors provide algorithms that find a primary path satisfying QoS requirements combined with a restoration topology. A restoration topology is a set of bridges, each of which circumvents a (different) part of the primary path. The proposed solution may violate the delay constraint for restoration paths, while the primary paths always satisfy the QoS constraints. One of the key contribution of this paper is the concept of adjusted delays, which allows existing path computation algorithms to be adapted in order to identify suitable restoration topologies.

In [GS98], W.D. Grover and D. Stamatelakis present the concept of p-cycle. In p-cycles, spare links are connected into cycles, but the method is different from self-healing rings because each preconfigured cycle contributes to the restoration of more failure scenarios than a ring can. One p-cycle can be used for the restoration of one span *on* the cycle (like self-healing rings), but also for the restoration of one span *off* the cycle if both ends of this span are located on the cycle. This reduces the amount of bandwidth that is consumed for restoration purpose, comparing to self-healing rings, so that the authors claim that p-cycles provide ring-like restoration speed with mesh-like capacity usage. The concept of p-cycles can be applied in IP networks [SG00].

9.1.2 Structure of the chapter

We provide a solution for the problem of precomputed local rerouting paths where bandwidth is reserved for backup paths. One backup LSP protects one primary LSP against one specific failure (node or link). This scheme can protect important traffic aggregates that require rapid restoration in case of failure and

cannot wait for a global restoration. Typically, traffic with strong QoS requirements need this kind of guarantee. Local restoration requires the establishment of many protection LSPs for which the bandwidth has to be reserved. This can lead to a big waste of bandwidth if not done properly. The best way to reduce the bandwidth waste is to make the assumption that two resources (link or node) cannot fail at the same time. When a failure occurs in the network, we suppose that the preceding failure has been recovered[1]. This assumption allow us to share bandwidth between two backup LSPs that will never be active together because they protect different resources.

A first contribution (from Mélon et al. in [MBL03]) is an improved bandwidth sharing scheme. When a failure occurs in the network, the traffic that makes a detour frees some bandwidth on some part of the primary path. In some cases, this bandwidth can be used without additional cost by another backup LSP protecting the failed resource, as it is explained in detail in section 9.2. This can decrease the bandwidth cost of both local and end-to-end protection. In section 9.3, we present an in-depth description of our algorithm. Section 9.4 explains how the available bandwidth is computed on a link when preemption levels are used.

Then our second contribution is a method to compute the local backup paths in a decentralized manner. Indeed, to be able to share efficiently the bandwidth and so reduce the bandwidth cost of the protection, the entity which computes the backup paths has to know a certain amount of information, as we explain in section 9.5. The first (naive) idea is to flood all the information in the whole network using the TE (Traffic Engineering) extensions of the intra-domain routing protocol. As this approach is obviously not scalable, we propose an original method that is capable of sharing a large amount of bandwidth while being scalable. In section 9.6, we justify our design choice of section 9.5.

Simulation results in section 9.7 give numerical values of the bandwidth gain which can be achieved using our technique. Finally, we conclude in section 9.9.

9.2 Algorithm overview

Our algorithm offers improvements in two main areas compared to previously presented solutions: (1) reduction of the bandwidth consumed by both local and end-to-end protection schemes by improving bandwidth sharing and (2) a method to handle preemption levels when bandwidth sharing is used in the network. Moreover, we provide an efficient and scalable way for the computing nodes to obtain the essential information to optimize bandwidth sharing.

[1]This assumption is implicit in all the 1+1 or 1:1 protection schemes. Indeed, if we want to protect traffic against double failures in the network, we have to protect backup paths against failures as well.

9.2.1 Bandwidth sharing

9.2.1.1 Backup-Backup bandwidth sharing

As already explained, resource sharing is possible under the assumption that, at any given time, at most a single failure will occur in the network. If this assumption holds, two backup LSPs protecting two distinct nodes will never be activated together. If these backup LSPs use some common links, we can reserve on these links only the maximum bandwidth requirement of both LSPs instead of their sum. Figure 9.1 presents this scenario. This type of bandwidth sharing can greatly decrease the bandwidth that need to be reserved for backup paths and is used in all classical 1:1 protection schemes.

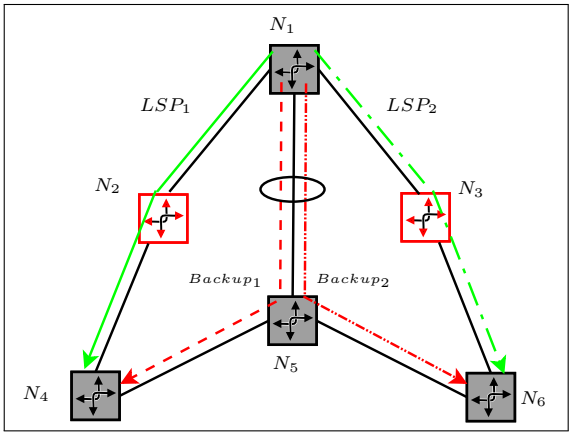

Figure 9.1: $Backup_1$ protects LSP_1 from failure of node N_2. $Backup_2$ protects LSP_2 from failure of node N_3. Since $Backup_1$ and $Backup_2$ will never be used simultaneously, they can share bandwidth on link $N_1 - N_5$.

9.2.1.2 Primary-Backup bandwidth sharing

It is possible to improve further the scheme if we consider that when a backup path is activated because of a failure, some bandwidth is not used any more on the primary path ([MBL03]). Indeed as soon as the failure is detected, the node responsible for the local backup will swap service and recovery paths. Very rapidly the circumvented links of the primary path will see their bandwidth consumption reduced. This bandwidth can thus be used by other backup LSPs protecting the same failed resource. Figure 9.2 details the situation.

9.2.1.3 Path computation

In our approach, path computation is completely decentralized and real-time. The LSP requests are received sequentially by the ingress nodes that compute and establish the LSPs one after the other. Consequently our scheme can combine easily with TE (Traffic Engineering) algorithms following the same decentralized philosophy. Of course, it can also combine with a centralized scheme.

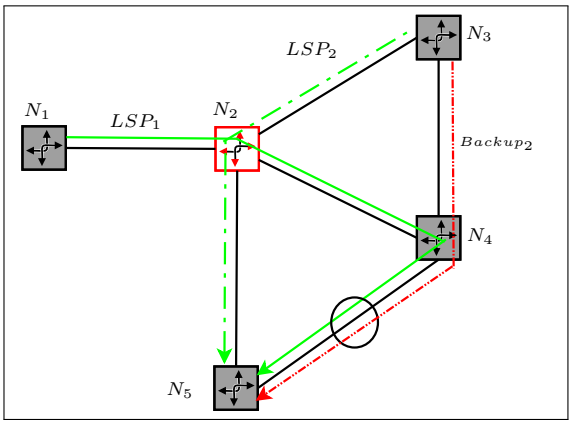

Figure 9.2: The two primary LSPs (LSP_1 and LSP_2) will fail together when N_2 fails. $Backup_2$, protecting LSP_2, can share bandwidth with LSP_1 on link $N4 - N5$. $Backup_1$, protecting LSP_1, is not shown on the figure.

In this work we will always assume without loss of generality that the primary LSP follows the shortest path according to a certain metric (usually a hop count). Other online constraint-based routing algorithms presented in section 2.2 could also be used. When the primary path is known, we compute the set of backup LSPs required to prevent any possible node failure along this path. If a backup path cannot be found under the node-failure assumption[2], we assume that only a link failure will occur and compute a new backup path. If it fails again, the request is rejected.

To use bandwidth efficiently, the path computation algorithm has to choose paths where the resource sharing is high. To compute the backup path we associate with each link a cost corresponding to the increment of bandwidth required if the backup LSP goes through the considered link. Dijkstra's algorithm is then used to compute the shortest path starting at the node preceding the protected node of the primary path towards the egress node. We stop the algorithm when it reaches a node that belongs to the primary path after the protected node.

9.2.2 Preemption levels

Preemption levels (see [BML03a]) are used to define some LSPs as being "more important" than others. When establishing an LSP, it can preempt the bandwidth reserved by LSPs having a lower preemption level. In case of failure, LSPs with a higher preemption level will also be restored first.

Handling preemption levels has no impact on the bandwidth sharing efficiency. However combining resource sharing and preemption levels in the same scheme requires some special care. This problem will be explained more extensively in section 9.4.

[2]Obviously it is impossible to protect the path against the failure of the egress node. However it is possible to protect the link between the penultimate node and the egress. The backup computed for this purpose only needs to be link-disjoint with the primary path.

9.3 In-depth description

For the clarity of the rest of this chapter we will first define a few terms and functions. In figure 9.3, we can see the differences between the link and the node protection.

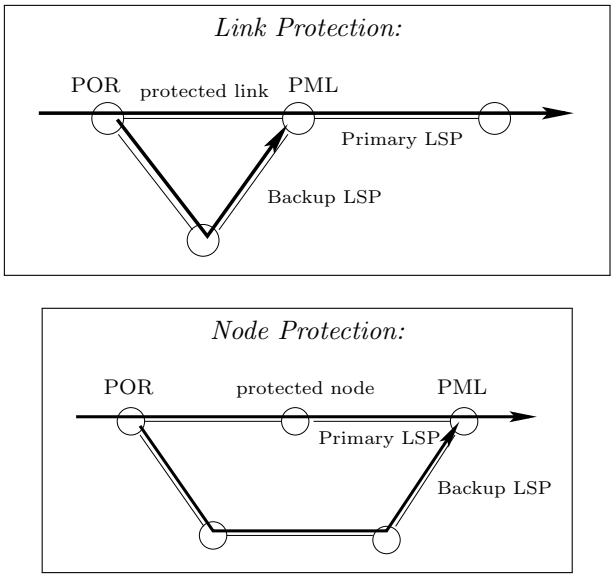

Figure 9.3: Link or Node protection

The following two notations come from [SMH03].

Definition 1. *The Point Of Repair (POR) is the LSR (Label Switching Router) where the switching is done between the primary and the backup path at the moment of the failure.*

Definition 2. *The Path Merge LSR (PML) is the LSR where the backup path merges with the primary path.*

A network is represented by a multi-valued graph $\mathcal{G} = (\mathcal{X},\mathcal{U})$ where $\mathcal{X}$ is a set of nodes and $\mathcal{U}$ a set of directed links between these nodes. Each link $L_{ij} \in \mathcal{U}$ between nodes $N_i \in \mathcal{X}$ and $N_j \in \mathcal{X}$ is associated with a set of values[3]:

- C_{ij} : the capacity of the link.
- R_{ij} : the total bandwidth reserved on the link.
- $R_{ij}[p]$: the total bandwidth reserved at preemption level p[4].
- $P_{ij}[p]$: the total bandwidth reserved at preemption level p for primary LSPs.

[3]Note that each node does not keep track of and store all of the presented information, as presented later on figure 9.9.

[4]In section 9.4 we explain how to compute $R_{ij}[p]$ such that $R_{ij} = \sum_{p=0}^{P-1} R_{ij}[p]$. Note that we do not need $R_{ij}[p], \forall p$ to compute R_{ij} (see equation 9.3).

- $P_{ij} = \sum_{p=0}^{P-1} P_{ij}[p]$.
- $B_{ij}(L_{kn})[p]$: the total bandwidth used by backup LSPs at preemption level p in case of failure of link L_{kn}.
- $B_{ij}(L_{kn}) = \sum_{p=0}^{P-1} B_{ij}(L_{kn})[p]$.
- $B_{ij}(N_k)[p]$: the total bandwidth used by backup LSPs at preemption level p in case of failure of node N_k.
- $B_{ij}(N_k) = \sum_{p=0}^{P-1} B_{ij}(N_k)[p]$.
- $F_{ij}(L_{kn})[p]$: the total bandwidth freed by primary LSPs at preemption level p in case of failure of link L_{kn}.
- $F_{ij}(L_{kn}) = \sum_{p=0}^{P-1} F_{ij}(L_{kn})[p]$.
- $F_{ij}(N_k)[p]$: the total bandwidth freed by primary LSPs at preemption level p in case of failure of node N_k.
- $F_{ij}(N_k) = \sum_{p=0}^{P-1} F_{ij}(N_k)[p]$.

P is the number of preemption levels. In a practical implementation the source node of each link is responsible for maintaining this set of values up-to-date. Now, let us have a look at figure 9.4. This figure represents the bandwidth utilization of a fixed link L_{ij}. We can see that the reserved bandwidth for the primary LSPs (P_{ij}) is the sum of the reserved bandwidth for each of the primary LSPs using this link. On the figure, we can see that there are four primary LSPs that pass on link L_{ij} (b_1, b_2, b_3 and b_4 are the values of the bandwidth reserved for the four primary LSPs). The bandwidth reserved for the primary LSPs is thus not shared at all. On the other hand, the bandwidth reserved for the different backup LSPs ($R_{ij} - P_{ij}$) is not the sum of the reserved bandwidth for each of the backup LSPs that pass on the considered link. Indeed, the bandwidth is shared between the backup LSPs that protect different resources. X_1, X_2, X_3 and X_4 are four resources (link or node) that are supposed not to fail at the same time. The total reserved bandwidth on link L_{ij} is R_{ij}. The free bandwidth on link L_{ij} is $C_{ij} - R_{ij}$.

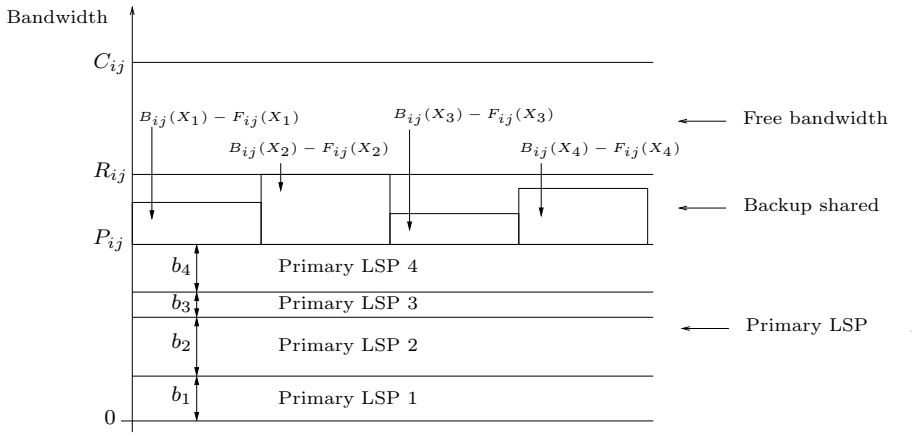

Figure 9.4: Repartition of the bandwidth on the link L_{ij}

From figure 9.4 and above definitions, we have:

$$B_{ij}(N_k)[p] = \sum_{\forall m: L_{mk} \in \mathcal{U}} B_{ij}(L_{mk})[p] \tag{9.1}$$

$$F_{ij}(N_k)[p] = \sum_{\forall m: L_{mk} \in \mathcal{U}} F_{ij}(L_{mk})[p] \tag{9.2}$$

$$R_{ij} = P_{ij} + \max\left(0, \max_{L_{kn} \in \mathcal{U}} \left(B_{ij}(L_{kn}) - F_{ij}(L_{kn})\right), \max_{N_k \in \mathcal{X}} (B_{ij}(N_k) - F_{ij}(N_k))\right) \tag{9.3}$$

Equations 9.1 and 9.2 express that a node failure is equivalent to the failure of all its incoming links. We consider incoming links because it is the same upstream nodes that will activate the same backup paths in case of a node failure or in case of the failure of all its incoming links. Note that in equation 9.3 we have to consider the maximum over all possible link failure scenarios even if we are protecting against node failure because it is not mathematically guaranteed that the worst case bandwidth consumption will be obtained when faced with a node failure. Indeed consider the failure of link "N_3-N_2" on figure 9.2. In this scenario, both Backup_2 and LSP_1 will be used simultaneously while it is not the case if node "N_2" goes down. Of course, in most practical situations the worst case will be a node failure.

An LSP request is composed of:

- the source or ingress node: src ;
- the destination or egress node: dst ;
- the required bandwidth[5]: bw ;
- the priority: p.

9.3.1 Link state management

Each node N_i has to maintain and update the link state information for all the links that originate at node N_i. This section will explain how the values of P_{ij}, B_{ij} and F_{ij} have to be updated for all links L_{ij} when a primary or backup LSP is established. Let $\mathcal{P}_{pr}$ be a primary LSP of preemption level p and required bandwidth bw. The path of this LSP is the ordered set $\mathcal{P}_{pr} = \{N_{y_0}, N_{y_1}, \ldots, N_{y_n}\}$, as shown on figure 9.5. Let $\mathcal{P}_{bu}$ be a backup LSP protecting the previously presented primary LSP. The path of this backup LSP is the ordered set $\mathcal{P}_{bu} = \{N_{x_0}, N_{x_1}, \ldots, N_{x_n}\}$. Let s be the index for which $N_{y_s} = N_{x_0}$ (= POR), and e be the index for which $N_{y_e} = N_{x_n}$ (= PML) i.e. the node where

[5]In this work we assume that a single value defines the bandwidth required by each LSP. In a DiffServ context this corresponds to using L-LSPs or E-LSPs with a single OA (Ordered Aggregate). Extensions of the presented algorithms to handle E-LSPs with multiple OAs is straightforward. Interested readers are invited to read [Fa02] for further information on how to combine DiffServ and (G)MPLS and for a definition of L-LSPs and E-LSPs.

primary and backup paths merge. The backup path protects node $N_{y_{s+1}}$ and link $L_{y_s y_{s+1}}$.

When the primary LSP is established, links $L_{ij} \in \mathcal{P}_{pr}$[6] must be updated according to:

$$P_{ij}[p] \leftarrow P_{ij}[p] + bw \tag{9.4}$$

When the backup LSP $\mathcal{P}_{bu}$ is established, all links $L_{ij} \in \mathcal{P}_{pr} \cup \mathcal{P}_{bu}$ must be updated according to:

$$B_{ij}(L_{y_s y_{s+1}})[p] \leftarrow B_{ij}(L_{y_s y_{s+1}})[p] + bw \qquad \text{if } L_{ij} \in \mathcal{P}_{bu} \tag{9.5}$$

$$F_{ij}(L_{y_s y_{s+1}})[p] \leftarrow F_{ij}(L_{y_s y_{s+1}})[p] + bw \qquad \text{if } (L_{ij} \in \mathcal{P}_{pr}) \wedge i < e \wedge j > s \tag{9.6}$$

The purpose of equation 9.6 is to free bandwidth on the primary path between the POR and the PML in case of failure. To apply equation 9.6, a backup LSP must also be signalled on part of the primary path (between the POR and the PML) in case of decentralized deployment. When $P_{ij}[p]$, $B_{ij}(L)[p]$ and $F_{ij}(L)[p]$ have been updated according to equations 9.4 to 9.6, $R_{ij}[p]$ can then be recomputed by means of a procedure described in section 9.4.

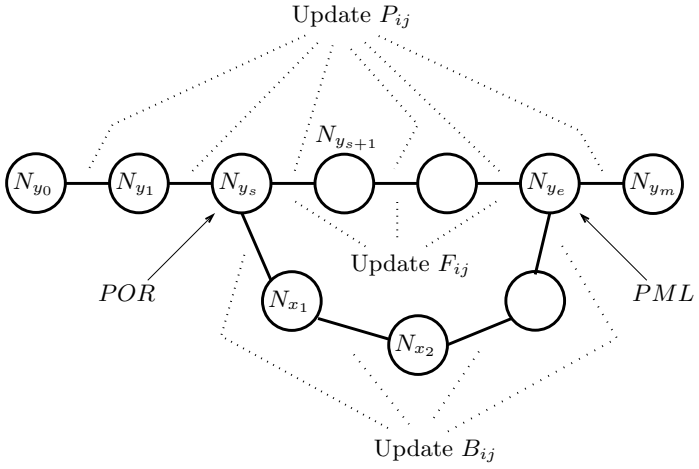

Figure 9.5: Primary and backup paths

9.3.2 Path computation

In this section we will explain the primary and backup paths computations. We will also highlight what kind of information is required to perform these computations. For the moment we assume that a centralized entity has access to all the link state information in the whole network. In section 9.5 we will explain how it is possible to distribute these computations.

[6]The notation $L_{ij} \in \mathcal{P}_{pr}$ denotes $\exists t : N_{y_t} = N_i \wedge N_{y_{t+1}} = N_j$.

9.3.2.1 Primary path computation

We assume that local protection will be required for important flows that are sensitive to delay and do not accept the delay of end-to-end protection in case of failure. For these flows we assume that the primary path has to be optimized on its own because it can have some strict delay constraints (the backup paths will be computed next). For example, we can use a Dijkstra's algorithm [W.59] to find the constraint shortest path from *src* to *dst*, considering a cost of one for all the links of the network (leading to a min hop path). Note that we can also rely on any other more suitable technique to compute the primary LSPs (e.g. those presented in section 2.2). We are aware that optimizing primary and backup paths together, instead of sequentially, could lead to a better global resource sharing. Indeed, our choice to take the primary path as a constraint for the backup paths computations limits our search to a subset of the whole state space. On the other hand, this combined method would lead to less optimized primary LSPs, which is a more severe shortcoming given that primary LSPs are used almost all the time.

The computed primary path can be described by an ordered set $\mathcal{P} = \{N_{x_0}, N_{x_1}, \ldots, N_{x_n}\}$ with $N_{x_0} = src$ and $N_{x_n} = dst$.

9.3.2.2 Backup paths computation

Given the primary path $\mathcal{P}$ and the node we try to protect $N_{x_{k+1}} \in \mathcal{P}$, we introduce $Inc_{ij}(N_{x_{k+1}}, bw)$ (resp. $Inc_{ij}(L_{N_{x_k}N_{x_{k+1}}}, bw)$) which represents the increase of R_{ij} when a backup LSP requiring bw units of bandwidth and protecting node $N_{x_{k+1}}$ (resp. link $L_{N_{x_k}N_{x_{k+1}}}$) uses link L_{ij}. We have $Inc_{ij}(*, bw) = R_{ij}^{'} - R_{ij}$ where $R_{ij}^{'}$ is the new reserved bandwidth obtained after the new LSP establishment. If $R_{ij}^{'} > C_{ij}$ then $Inc_{ij}(*, bw) = \infty$ (capacity constraint). R_{ij} and $R_{ij}^{'}$ can be calculated using equations 9.1, 9.2 and 9.3.

Each link L_{ij} is assigned a cost K_{ij} given by

- if we protect against failure of node $N_{x_{k+1}}$

```
if (i = N_{x_{k+1}} ∨ j = N_{x_{k+1}})
    K_ij = ∞
else if (Inc_ij(N_{x_{k+1}}, bw) = 0)
    K_ij = ε
else
    K_ij = Inc_ij(N_{x_{k+1}}, bw)
```

- if we protect against failure of link $L_{N_{x_k}N_{x_{k+1}}}$

```
if (i = N_{x_k} ∧ j = N_{x_{k+1}})
    K_ij = ∞
else if (Inc_ij(L_{N_{x_k}N_{x_{k+1}}}, bw) = 0)
    K_ij = ε
else
    K_ij = Inc_ij(L_{N_{x_k}N_{x_{k+1}}}, bw)
```

Dijkstra's algorithm is run from root N_{x_k} until the next marked node by the procedure is N^* with $N^* \in \{N_{x_{(k+1)}}, ..., N_{x_n}\}$[7]. If no valid node-disjoint path is found, then select link failure protection. If it fails once again, reject the request. We have introduced the small number ε which is used instead of zero, to favour the selection of the minimum hop path if all Inc_{ij} are null.

Needed information To compute the cost K_{ij} for all the links of the network, we need some information. We claim that R_{ij}, P_{ij}, $B_{ij}(X)$ and $F_{ij}(X)$ are sufficient to compute K_{ij}, if X is the resource we want to protect (link or node). Indeed, to compute the increment of bandwidth which would result from the establishment of the backup path on link L_{ij}, we must check whether the new value of $B_{ij}(X) - F_{ij}(X)$ is greater than the *old* value of $R_{ij} - P_{ij}$.

About optimality It is worth noting that our procedure is not optimal for two reasons. First of all, it is not a network-wide optimum. It is quite obvious because requests are treated one after the other. This means that choices made for any particular LSP will never be re-evaluated in the future. But this procedure is also not optimal at the LSP level, i.e. does not lead to find the set of LSPs minimizing the increase of bandwidth reservation. Indeed, all the backup LSPs (one for each node of the primary path) are calculated one after the other. Once again the sequentiality prevents the algorithm to find an optimal solution.

Despite being sub-optimal, our simulations have shown that this algorithm was a good heuristic. We also tested an enhanced version of the algorithm. In this version, once all the backup paths of one primary path are computed, we try to improve the sharing of each of them (by changing their path) with the knowledge of all the other backup paths until no more bandwidth gain is observed. In very rare cases it leads to a global improvement on the total bandwidth consumed at the network level. In all other cases it leads to no improvement at all or even to an increase in the bandwidth consumption.

Moreover the type of solution we propose is designed to be used in a dynamic environment where relatively small LSPs (compared to the links capacity) are added and removed permanently, following users needs. In this context, a form of statistical multiplexing makes the ordering of establishment less relevant.

9.4 Preemption levels aggregation

When combining both preemption levels and resource sharing we must be careful that $R_{ij}[p]$ must correctly reflect the amount of bandwidth which can effectively be preempted (if required) at each preemption level. Indeed, the amount of bandwidth assigned to each preemption level has to reflect the fact that by removing all LSPs at a given level a certain amount of bandwidth will be freed. For primary paths, we have to reserve at level p the sum of the bandwidth required by all LSPs at level p. The introduction of backup LSPs and bandwidth sharing makes things a bit more complex. Indeed when using protection, removing an LSP does not necessarily free any resource : if we recall equation 9.3,

[7] In case of node protection, the end of the backup tunnel cannot be $N_{x_{k+1}}$ because all its connected links have an infinite weight.

we see that a decrease of $B_{ij}(N)$ only has an impact on R_{ij} if node N is the one that maximizes the difference. The consequence is that the preemption of a given quantity of bandwidth will sometimes require that we tear down a set of LSPs whose total bandwidth is bigger than the required bandwidth. To do so the LSPs we try to establish must have a preemption level higher than all the LSPs in this set.

Important remark : In this work, preemption levels are numbered in decreasing order of priority. Level 1 is thus more important than level 2.

An example is given in tables 9.1 and 9.2. The bandwidth that must be reserved for backup LSP_1 and LSP_2 can be limited to $max(BW(\text{LSP}_1), BW(\text{LSP}_2)) =$ 10 Mbps because they protect two distinct nodes. A new LSP with preemption level 1 would only be able to preempt bandwidth from LSP_1. But despite the fact that LSP_1 requested 10 Mbps of bandwidth, removing it will only free 5 Mbps because of sharing.

LSP	Failure	Bandwidth	Preemption Level
1	N_x	10 Mbps	2
2	N_y	5 Mbps	1

Table 9.1: Sharing with preemption levels: LSPs

Preemption level	Bandwidth
$R_{ij}[1]$	5 Mbps
$R_{ij}[2]$	5 Mbps
R_{ij}	10 Mbps

Table 9.2: Sharing with preemption levels: $R_{ij}[p]$

As explained earlier preemption levels are used to give priority to certain LSP requests. If a link is completely filled then it is still possible to establish a new LSP through this link by preempting resources belonging to less important LSPs. But the bandwidth reserved on a link is the result of three terms (cf. equation 9.3) which are composed in different proportions for each preemption level.

The algorithm computing $R_{ij}[p]$ is composed of two phases. The first one consists of computing an intermediate result $G_{ij}(L)[p]$ and $G_{ij}(N)[p]$. The second phase computes $R_{ij}[p]$ using this result.

9.4.1 Phase 1

The value $G_{ij}(L)[p]$ (resp. $G_{ij}(N)[p]$) represents, up to preemption level p, the bandwidth that must be reserved on link L_{ij} to be able to forward traffic in case of failure of link L (resp. node N), in addition to the bandwidth that is reserved for primary LSPs.

$$G_{ij}(L)[p] \leftarrow \max\left(0, \sum_{k=0}^{p} B_{ij}(L)[k] - \sum_{k=0}^{p} F_{ij}(L)[k]\right) \quad \forall p, L : 0 \leq p < P, L \in \mathcal{U}$$

$$G_{ij}(N)[p] \leftarrow \max\left(0, \sum_{k=0}^{p} B_{ij}(N)[k] - \sum_{k=0}^{p} F_{ij}(N)[k]\right) \quad \forall p, N : 0 \leq p < P, N \in \mathcal{X}$$

The algorithm is based on the following idea: we do not need to reserve extra bandwidth at level p if, up to that level, a sufficient amount of bandwidth will be freed by the failure we consider. However we should note that $G_{ij}(X)[p]$ (X being either a node or a link) can never be negative even if $\sum F_{ij}(X)[p] > \sum B_{ij}(X)[p]$ because it would mean we have to "unreserve" bandwidth that is used by active primary paths (recall that $F_{ij}(X)[p]$ is already reserved).

Figure 9.6 shows the situation. Up to level 1, a failure will free more bandwidth than needed by the backup LSPs. For this reason $G_{ij}(X)[0] = G_{ij}(X)[1] = 0$. For $p \geq 2$, $\sum_{k=0}^{p} B_{ij}(X)[k] - \sum_{k=0}^{p} F_{ij}(X)[k] > 0$. The values of $G_{ij}(X)[2]$, $G_{ij}(X)[3]$ and $G_{ij}(X)[4]$ are represented graphically on the figure.

$G_{ij}(X)[p]$ is the balance up to level p between the required backup bandwidth and the freed primary bandwidth. If $G_{ij}(X)[p_0] > 0$ for a particular p_0, this means that we must add a new reservation of bandwidth at level p_0 to correct the difference. If we assume that the correction has already been done for all $p < p_0$, the new reservation at level p_0 must consist of $G_{ij}(X)[p_0] - G_{ij}(X)[p_0 - 1]$ units of bandwidth.

This reasoning is only for a particular failure. As any node in the network can fail, we have to define a new vector $M_{ij}[p]$ accounting for the maximum difference between the total bandwidth required and freed considering all possible failures. This is the purpose of phase 2.

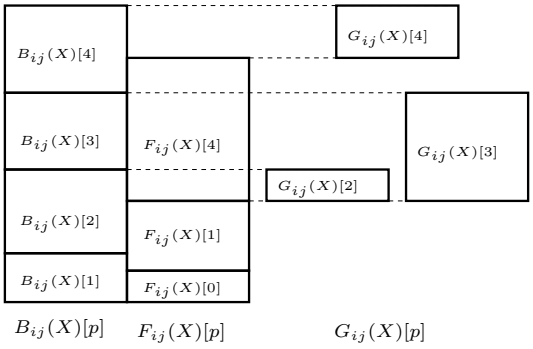

Figure 9.6: Preemption level selection

9.4.2 Phase 2

We introduce the vector $M_{ij}[p]$ given by :

$$M_{ij}[p] \leftarrow \max\left(\max_{L\in\mathcal{U}}\left(G_{ij}(L)[p]\right), \max_{N\in\mathcal{X}}\left(G_{ij}(N)[p]\right)\right) \quad \forall p : 0 \leq p < P$$

This vector plays the same role as $G_{ij}(X)[p]$ but at the network-wide level. Now that we have such a failure independent value we can compute :

$$\begin{cases} R_{ij}[0] \leftarrow P_{ij}[0] + M_{ij}[0] \\ R_{ij}[p] \leftarrow P_{ij}[p] + M_{ij}[p] - M_{ij}[p-1] \\ \qquad\qquad\qquad\qquad \forall p : 0 < p < P \end{cases}$$

This formula reflects the computations we made in the example of tables 9.1 and 9.2. It should be pointed out that the difference $M_{ij}[p] - M_{ij}[p-1]$ can be negative which looks a bit surprising at first. Indeed it means we have to reserve less bandwidth at level p than the sum of the bandwidth requirements of all primary LSPs. In fact this just means that a certain amount of bandwidth initially reserved at level p has been upgraded to level $p_0 < p$ to be aggregated with backup LSPs.

9.5 The signalling problem

We will now study how the nodes can obtain the information they need to compute all the paths in a decentralized scheme. In sections 9.3.2.1 and 9.3.2.2, we saw which information is needed to compute primary paths, backup paths and the reserved bandwidth on a link. In this section, we will understand where this information is needed and the solution we propose to achieve our objective, i.e. to stay scalable while achieving optimal bandwidth sharing.

9.5.1 Where must the information be available?

First, we can say that the information is needed where the computations are made. If all the computations are made by a centralized server, the solution is simple: it computes all the (primary and backup) paths and thus it knows all the LSPs of the network. The objective of this section is to discuss how our proposal can be decentralized.

The **computation of the primary path** is made by the ingress node. Indeed, this node is the most appropriate because it receives the request for the creation of the LSP.

The **computation of the reserved bandwidth on a link** ($R_{ij}[p], \forall p$) is made by the node (N_i) immediately upstream of the link. This computation influences the admission control of a new LSP (primary or backup) on that link.

For the **computation of the backup path**, different solutions are possible. This computation can be made by:

- The ingress of the primary LSP. In this case, the ingress computes all the backup paths that protect all the links and nodes we want to protect on

the primary path. After all the computations, it forwards these backup paths to the nodes on the primary path which will establish them.

- The POR. In this case, this is the node immediately upstream of the link or node we want to protect (the POR) that computes the backup path. It is also this node that establishes the backup path.

- The node to be protected or the node immediately downstream of the link to be protected. This node computes the backup path and sends it to the POR which establishes it.

In section 9.6, we compare these three solutions. We have chosen to use the third one which appears to be excellent. Indeed, as we see in that section, the bandwidth cost is minimum and we can extend RSVP to support this solution without requiring any additional signalling protocol.

9.5.2 Establishment of the LSPs

In this paragraph, we present a preview of our signalling solution. The computation of the backup paths is distributed between all the nodes of the primary path. They are computed when the RESV message[8] of the primary path is sent by the egress back to the ingress node. At this time, each node, one after the other, will compute one backup path protecting itself or its upstream link[9]. It will send the computed backup path to the upstream node. This node will establish this backup path, compute a new backup path protecting itself or its upstream link and send it to its upstream node. This operation is repeated until the ingress node of the primary path is reached. In addition to the backup path, each node must send to its upstream node the union of the links already used by previously computed backup paths, as this information will be used by upstream nodes to compute other backup paths.

Each node keeps in memory some fields of the RSVP messages it transmits. Since RSVP is a soft state protocol, the path is refreshed regularly. If the path is not refreshed, the LSP (and its associated information) disappears.

Each node which computes a backup path keeps this knowledge in memory. Besides, only a very limited amount of information needs to be flooded in the link-state routing protocol, e.g. OSPF(-TE).

9.5.2.1 Establishment of a primary LSP

Each node on the path of the LSP decides whether to accept or reject the LSP. This decision is made considering the reserved bandwidth of the downstream link (capacity check). Obviously, to decide whether to accept or reject the LSP on a downstream link, the RSVP message must contain a flag specifying that it is a primary LSP. Once it is accepted, the node stores the value of the LSP's bandwidth (which updates the value of $P_{ij}[p]$).

[8] This is a message of the RSVP protocol ([AHX01]), which is used to establish LSPs in MPLS networks.

[9] We compare our technique to other possible solutions in section 9.6.

9.5.2.2 Establishment of a backup LSP

The same kind of admission control is also needed at the establishment of a backup LSP. As with primary paths, a flag of the RSVP message must specify the type of the LSP (i.e. backup in this case). Each node N_i has in memory the present value of $P_{ij}[p]$, $B_{ij}(X)[p]$, $\forall_j, X, p$. Indeed, N_i can compute these values thanks to the primary and backup RSVP messages seen at the establishment of the LSPs. To take primary-backup sharing into account, N_i also needs to know $F_{ij}(X)[p]$. For this purpose, we propose to forward a specific message on all the nodes of the primary path between the POR and the PML. We will call this message the $F_{ij}\_message$. This message must specify the primary LSP to which it is related and the resource (X) which is protected.

9.5.3 Computation of the LSPs

If we follow the procedure described in section 9.5.2, each node N_i is able to compute the primary reserved bandwidth ($P_{ij}[p], \forall p$) and the total reserved bandwidth ($R_{ij}[p], \forall p$) on all the links that originate at N_i. We propose that each node N_i floods $P_{ij}[p]$ and $R_{ij}[p], \forall p$ in the LSAs[10] of OSPF-TE together with the capacity (C_{ij})[11]. Thanks to this flooding, every node in the network knows $C_{mn}, R_{mn}[p]$ and $P_{mn}[p]$ $\forall p$ and $\forall_{mn} \mid L_{mn} \in \mathcal{U}$, i.e. for all the links of the network. Notice that we flood in the network the same amount of information than the partial information scheme of [KL01].

9.5.3.1 Computation of the primary LSPs

All the nodes know the free bandwidth of all the links. So all the nodes, including the ingress node, are able to compute primary LSPs.

9.5.3.2 Computation of the backup LSPs

As we have seen in section 9.3.2.2, a node which computes a backup path has to know some information about other LSPs protecting the same resource. Our idea is the following: "*If it is always the same node that computes all the backup paths protecting a certain resource, this node knows almost all he has to know to compute them!*" So, we propose to associate with each resource a dedicated node which computes the backup LSPs protecting it. The information obtained by the backup path computations will be used for future backup path computations.

In our proposal, for a node N, the dedicated node is N itself, and for a link L_{ij}, the dedicated node is N_j. We can see on figure 9.7 how the POR can ask for the computation of a path. Arrow 1 means: *"Compute for me a path protecting you, or similarly, protecting this link between me and you."* The protected node computes a backup path and sends it back to the POR (arrow 2), which establishes it. We propose to include this message exchange in the RSVP protocol (it is theoretically possible). The arrow 1 message is included

[10] LSA stands for Link State Advertisement.

[11] OSPF-TE already floods the capacity C_{ij} and the free bandwidth $C_{ij} - R_{ij}$ ([KKY03]).

Information	Obtained by	Exported
$P_{ij}[p]$	RSVP-TE of primary LSPs	Flooded with OSPF-TE
$P_{mn}[p],\ \forall m \neq i$	OSPF-TE	Kept locally
$B_{ij}(L_{mn})[p],\ \forall n \neq i$	RSVP-TE of backup LSPs	Kept locally to compute $R_{ij}[p]$
$F_{ij}(L_{mn})[p],\ \forall n \neq i$	$F_{ij}\_message$	Kept locally to compute $R_{ij}[p]$
$R_{ij}[p]$	Computation using $P_{ij}[p]$, $B_{ij}(L_{mn})[p], F_{ij}(L_{mn})[p]$	Flooded with OSPF-TE
$R_{mn}[p],\ \forall m \neq i$	OSPF-TE	Kept locally
$B_{mn}(L_{ki})[p]$	Computation of backup paths	Kept locally
$F_{mn}(L_{ki})[p]$	Computation of backup paths	Kept locally
$B_{mn}(L_{kj})[p],\ \forall j \neq i,\ \forall m \neq i$	Not needed	
$F_{mn}(L_{kj})[p],\ \forall j \neq i,\ \forall m \neq i$	Not needed	

Table 9.3: Database details at node N_i

as an object in the PATH message of the primary LSP and the arrow 2 message is included in the RESV message (see example in section 9.5.5).

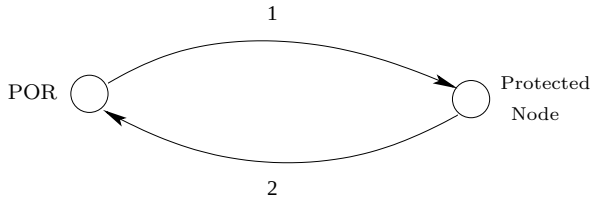

Figure 9.7: Example of message exchange

With this scheme, each node N_i knows all the backup paths of the network protecting itself and its incoming links because it has computed these backup paths. More formally, each node N_i knows: $B_{mn}(L_{ji})[p]$ and $F_{mn}(L_{ji})[p]$, $\forall p$, $\forall_{m,n,j} \mid L_{ji} \in \mathcal{U}$ and $L_{mn} \in \mathcal{U}$; and thus $B_{mn}(N_i)[p]$ and $F_{mn}(N_i)[p]$, $\forall p$ $\forall_{mn} \mid L_{mn} \in \mathcal{U}$ (see equations 9.1 and 9.2).

9.5.3.3 Data flow summary

Table 9.3 presents the structure of the information database at node N_i, $\forall_j \mid L_{ij} \in \mathcal{U}$, $\forall_k \mid L_{ki} \in \mathcal{U}$, $\forall_{mn} \mid L_{mn} \in \mathcal{U}$, $\forall p$. This table shows how N_i obtains information and how it is exported. Figure 9.8 shows the same information, but in a more convenient way. Figure 9.9 shows the part of the whole $B_{xx}(L_{xx})[p]$ table which is kept at each node. If M is the number of links in the network and K is the number of neighbours of node N_i then the size of the whole table would be M^2. Out of this table, only $K(2M-K)$ values are stored locally and $(M-K)^2$ are not used at all. Besides, none are flooded.

9.5.4 Simplification of signalling

The transmission of the $F_{ij}_message$s can be a problem. Indeed, they require an additional protocol on the primary path between the POR and the PML.

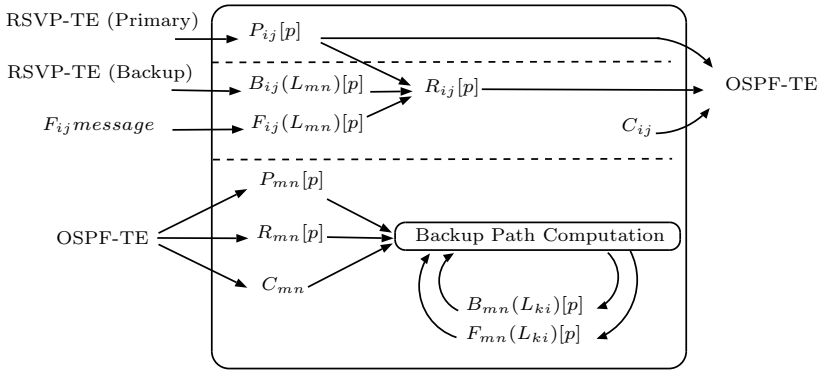

Figure 9.8: Data flow at node N_i

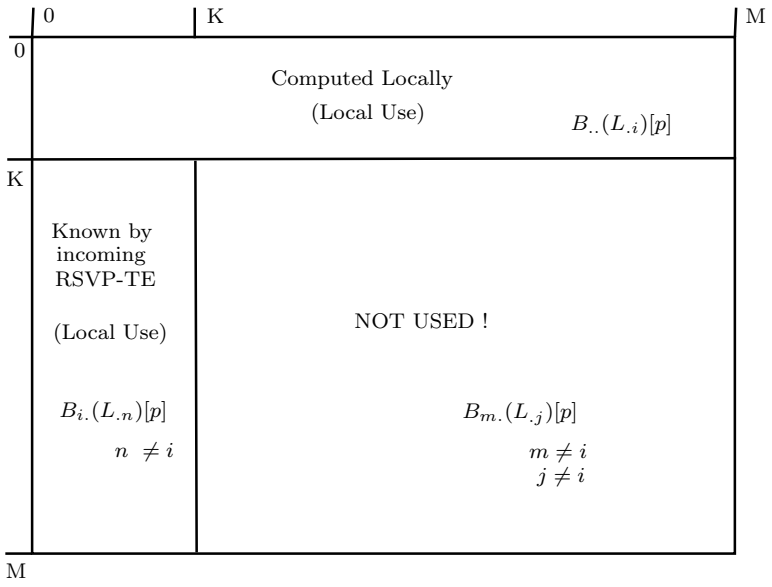

Figure 9.9: Information size

But in fact, we can see that we do not actually need them. Indeed, we can put them in primary RSVP refresh PATH messages. The only drawback of this method is that this introduces an additional delay (we must wait for the next PATH refresh message).

Doing so, all the nodes between the POR and the PML regularly send primary PATH refresh messages which contain $F_{ij}\_objects$. When a backup path is closed, these nodes stop sending the $F_{ij}\_objects$ in the primary PATH refresh messages. Doing so, these nodes (including the protected node) can update their database.

9.5.5 A simple example

In this section, we will show an example to clarify the explanations. We consider the topology of figure 9.10. In order to avoid complex notation, we will not mention preemption levels in this example. This does not remove any generality to our proposal. Implicitly, all the mentioned values are specified for the specific preemption level of the requested LSP.

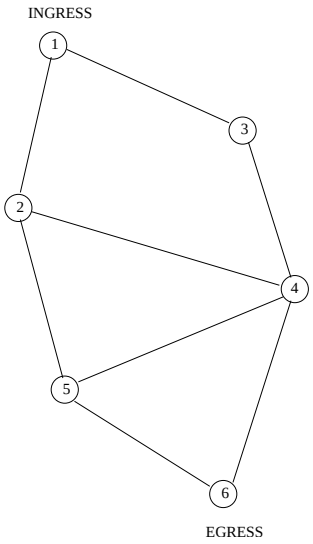

Figure 9.10: Topology of the example

Node 1 receives a request for the establishment of an LSP of b units of bandwidth from node 1 to node 6. Node 1 computes a primary LSP. Node 1 knows the free bandwidth on all the links because they have been flooded by the extended routing protocol (e.g. OSPF-TE). The computed path is $N_1 \rightarrow N_2 \rightarrow N_5 \rightarrow N_6$. Once the primary path is computed, node 1 establishes it. It sends a PATH message to node 2 (see figure 9.11a). In this PATH message, an RSVP object asking for a local restoration is added. Node 2 accepts the request and forwards the PATH message to node 5 (figure 9.11b). Node 5 accepts the request and forwards the PATH message to node 6 (figure 9.11c). Each node on the primary path has seen the PATH message so it knows that a protection LSP must be computed.

Node 6 computes a backup path that protects itself. As it is not possible, it computes a path protecting link L_{56}: $N_5 \rightarrow N_4 \rightarrow N_6$. Node 6 sends this path to node 5 as an object in the RESV message (figure 9.11d). Node 5 also computes a backup path that protects itself: $N_2 \rightarrow N_4 \rightarrow N_6$. It sends this path to node 2 with the RESV message. At the same time, it can establish the backup LSP which protects link L_{56} (figure 9.11e)[12]. Now, it is node 2 that receives the RESV message. This node computes a backup path that protects itself: $N_1 \rightarrow N_3 \rightarrow N_4 \rightarrow N_6$. Node 2 sends this path to node 1 with the RESV message. At the same time, node 2 establishes the backup path protecting node 5 (figure 9.11f). In this case, as the protected part of the primary LSP is greater than 2 links, node 2 has to send an $F_{ij}\_message$ to node 5. This message indicates that the failure of link L_{12} will free the primary bandwidth between node 1 and node 6. We can remark that although there are three backup paths, there is only one $F_{ij}\_message$ that must propagate in the network. Now, node 1 receives the RESV message from node 2 meaning that the primary LSP is established. Finally, node 1 establishes the backup path protecting node 2 (figure 9.11g).

We will now study in more detail when nodes store information in their memory. We can follow in table 9.4 which information is known at which moment by which nodes. For example, the first three lines show that when the PATH message of a primary path is forwarded by a node, this node stores this information. The fourth line shows that when a node computes a backup path, this node will keep a piece of information about this backup path. The fifth and sixth lines show that the nodes on the path of a backup LSP also store some information.

Step	**Node**	**Updated information**	**Obtained from**
a	1	P_{12}	RSVP (primary)
b	2	P_{25}	RSVP (primary)
c	5	P_{56}	RSVP (primary)
d	6	$B_{54}(L_{56})$ $B_{46}(L_{56})$ $F_{56}(L_{56})$	Local BP computation
e	5	$F_{56}(L_{56})$ $B_{54}(L_{56})$	RSVP (backup)
	4	$B_{46}(L_{56})$	RSVP (backup)
	5	$B_{24}(L_{25})$ $B_{46}(L_{25})$ $F_{25}(L_{25})$ $F_{56}(L_{25})$	Local BP computation
f	2	$F_{25}(L_{25})$ $B_{24}(L_{25})$	RSVP (backup)
	4	$B_{46}(L_{25})$	RSVP (backup)
	2	$B_{13}(L_{12})$ $B_{34}(L_{12})$ $B_{46}(L_{12})$ $F_{12}(L_{12})$ $F_{25}(L_{12})$ $F_{56}(L_{12})$	Local BP computation
g	5	$F_{56}(L_{12})$	$F_{ij}\_message$
	1	$F_{12}(L_{12})$ $B_{13}(L_{12})$	RSVP (backup)
	3	$B_{34}(L_{12})$	RSVP (backup)
	4	$B_{46}(L_{12})$	RSVP (backup)

Table 9.4: Evolution of the information transmission

In table 9.5, we can see which information is known by each node at the end of the process. It is the same information as in table 9.4, but sorted by node and type. We can see that this information is in agreement with the information of table 9.3 and figure 9.8.

[12]For reason of clarity, we do not show the PATH and RESV messages used for the estab-

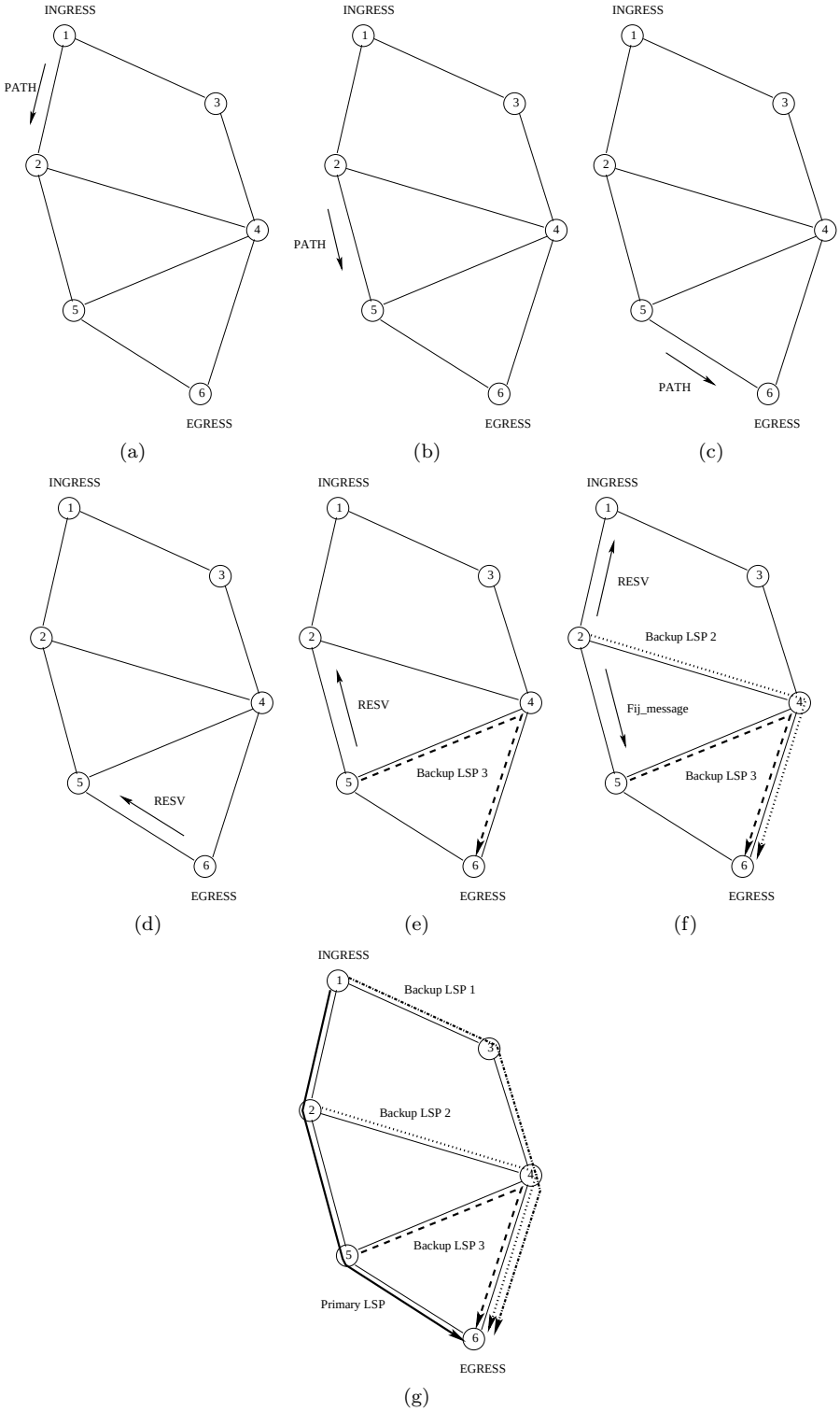

Figure 9.11: Example details

Node	Type	Information		
1	P_{ij}	P_{12}		
	$B_{ij}(L)$	$B_{13}(L_{12})$		
	$F_{ij}(L)$	$F_{12}(L_{12})$		
2	P_{ij}	P_{25}		
	$B_{ij}(L)$	$B_{24}(L_{25})$		
	$F_{ij}(L)$	$F_{25}(L_{25})$		
	$B_{mn}(L_{ki})$	$B_{34}(L_{12})$	$B_{46}(L_{12})$	$B_{13}(L_{12})$
	$F_{mn}(L_{ki})$	$F_{12}(L_{12})$	$F_{25}(L_{12})$	$F_{56}(L_{12})$
3	$B_{ij}(L)$	$B_{34}(L_{12})$		
4	$B_{ij}(L)$	$B_{46}(L_{56})$	$B_{46}(L_{25})$	$B_{46}(L_{12})$
5	P_{ij}	P_{56}		
	$B_{ij}(L)$	$B_{54}(L_{56})$		
	$F_{ij}(L)$	$F_{56}(L_{56})$	$F_{56}(L_{12})$	$F_{56}(L_{25})$
	$B_{mn}(L_{ki})$	$B_{24}(L_{25})$	$B_{46}(L_{25})$	
	$F_{mn}(L_{ki})$	$F_{25}(L_{25})$		
6	$B_{mn}(L_{ki})$	$B_{54}(L_{56})$	$B_{46}(L_{56})$	
	$F_{mn}(L_{ki})$	$F_{56}(L_{56})$		

Table 9.5: State of the tables of the nodes

9.6 Comparison to other possible signalling schemes

We will study in detail the three different solutions of section 9.5.1 and justify our choice. They differ in the node that computes the backup LSPs.

9.6.1 Presentation of the three possible locations for backup path computation

9.6.1.1 Computation by the ingress of the primary LSP

It is the ingress of the primary LSP that computes the backup paths protecting the links and nodes of the primary path. We consider two propositions to achieve this goal. The first proposition is that each node floods all the information ($B..(L..)$ and $F..(L..)$) in the whole network. Doing so, every node of the network (including the ingress node) can compute the backup paths. But this proposition is really not scalable.

The second proposition is that each node keeps the information concerning itself and sends it to the ingress node. Thus, firstly, the ingress node computes the primary LSP and establishes it. Secondly, each node on the primary path sends the information it owns to the ingress to allow it to compute the backup paths. When the backup paths are computed, the ingress sends each POR the backup path it has to establish. Furthermore, the POR forwards the backup path to the protected node so that it can keep this information in memory. The protected node has to be aware of this information because it will send it to the ingress of the future primary LSPs passing on it.

lishment of the backup LSPs.

With this scheme, each node N_i knows all the backup paths protecting itself, all the backup paths protecting the incoming links and all the backup paths using itself. Formally, this information is: $B_{mn}(L_{ji})$ and $F_{mn}(L_{ji})$, $\forall_{mn} \mid L_{mn} \in \mathcal{U}, \forall_j \mid L_{ji} \in \mathcal{U}$ and thus $B_{mn}(N_i)$ and $F_{mn}(N_i)$, $\forall_{mn} \mid L_{mn} \in \mathcal{U}$ (see equations 9.2 and 9.1).

9.6.1.2 Computation by the POR

It is the POR which computes the backup path. The POR is the node immediately upstream of the resource to protect. In case of link protection, there is only one node per protected link that can be the POR for the backup LSP. On the other hand, in case of node protection, there are multiple nodes that can potentially be a POR for a backup LSP protecting a particular node. Indeed, each neighbour of the node can be the POR. So, every potential POR must store the information about the node. Thus, after having established a backup LSP (see figure 9.12), the POR sends the new information to the protected node (arrow 1) which forwards it to all its neighbours except the POR (arrows 2).

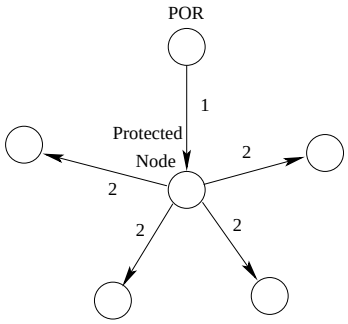

Figure 9.12: Example of exchange of messages in the case 9.6.1.2

With this scheme, each node i knows ($\forall k$ which is i or a neighbour of node i): $B_{mn}(L_{jk})$ and $F_{mn}(L_{jk})$, $\forall_{mn} \mid L_{mn} \in \mathcal{U}, \forall_j \mid L_{ji} \in \mathcal{U}$ and thus $B_{mn}(N_k)$ and $F_{mn}(N_k)$, $\forall_{mn} \mid L_{mn} \in \mathcal{U}$ (see equations 9.2 and 9.1).

9.6.1.3 Computation by the node to protect

It is the solution we have chosen. This has already been explained in section 9.5.3.2.

9.6.2 Evaluation of the performance of the three solutions

In this section, we will estimate the cost of the three solutions. For these estimations, we need to introduce some new notations:

- l is the number of nodes on the primary path
- $x =\mid \mathcal{X} \mid$ i.e. the number of nodes in the network
- $u =\mid \mathcal{U} \mid$ i.e. the number of links in the network

- s is the mean size of the backup paths
- t is the mean degree of the nodes, i.e. the number of neighbours of the node
- C_1 and C_2 are constant values representing the size in bytes of the representation of (respectively) one record in the database and the identifier of one node.

The bandwidth cost of a message is computed this way:

size_of_the_message * *number_of_links_crossed_by_the_message.*

9.6.2.1 First solution

In the first solution, the ingress must get the information from all the nodes of the primary path. The information for one node N_i is $\begin{cases} B_{mn}(N_i) - F_{mn}(N_i) \\ B_{mn}(L_{ji}) - F_{mn}(L_{ji}) \end{cases}$, $\forall m, n$. N_j is supposed to be the node preceding N_i on the primary path. Thus, the size of the information to transmit to the ingress node[13] for each node on the primary path is $2 * u * C_1$. The information of the egress node has to cross $(l-1)$ links to reach the ingress, but the information of the node just downstream the ingress node has to cross just one link to reach it. The total number of links crossed by the information of one node is: $1+2+...+(l-1) = \sum_{k=1}^{l-1} k = \frac{(l-1)*l}{2}$. The total cost is $\frac{2*u*C_1*(l-1)*l}{2}$ for all the messages that go from each node of the primary path to the ingress node.

When the ingress node has computed all the backup paths, it has to send them to the PORs because it is the PORs that will establish them. The cost of one path message is $s * C_2$. So, with similar arguments like above, the total cost of these messages is $\frac{s*C_2*(l-1)*l}{2}$.

In conclusion, the total bandwidth cost of this first solution is $\frac{(s*C_2+2*u*C_1)*(l-1)*l}{2}$. Furthermore, this solution requires an additional signalling for every node to send its information to the ingress and for the ingress to send the computed backup path to the nodes on the primary path.

9.6.2.2 Second solution

The POR can compute the backup path. After having established the backup path, it has to transmit the computed path to the downstream node. This node will in turn transmit this information to its neighbours (see figure 9.12). The bandwidth cost of this operation is $s * t * C_1$. This operation requires an additional signalling.

9.6.2.3 Third solution

The POR asks to the downstream node to compute the backup path protecting it. After the computation, the protected node sends the path to the POR which

[13]Here, we do not take into account the fact that in case of lightly loaded networks, a high number of B and F components may be equal to zero.

establishes it. The bandwidth cost of this operation is $s * C_2$. This operation may require an additional signalling protocol. But as we have seen, we can extend RSVP to support this scenario.

9.7 Simulation results

This simulation section is composed of two parts. The first part contains a deep analysis of the results of the simulation of our algorithm on a first topology. In the second part, we briefly present some results obtained on three other topologies.

9.7.1 Detailed results

For this part of the simulation, we have used a randomly generated topology which was composed of 50 nodes and 102 full-duplex links. Among the 50 nodes, 30 were chosen to act as border routers. We assigned to each ingress-egress pair a probability. The topology used has been "perfectly engineered" thanks to a *Generalized maximum concurrent flow* algorithm. By "perfectly engineered", we mean that a load of 100% is reachable throughout the network if the ingress-egress probabilities are respected and the LSP-bandwidth are infinitesimal. This has been done because we realized it was useless to engineer the traffic on a network with engineering inconsistencies such as huge links following very small ones (they can only reach a very small relative load). And indeed, the *Generalized maximum concurrent flow* approximation we used is close to the behaviour used by some network engineers we have met (*e.g. "double up the link capacity when it reaches 50% of load"*). Network engineering in the context of fault protection is still a very active domain of research (cf. [LTS01, LT01]).

The most important value when designing a restoration scheme is the "cost" of such a protection in terms of additional bandwidth reserved for backup LSPs. We will call it "network oversubscription" and represent it by γ. It is given by:

$$\gamma = \frac{\sum_{L_{ij} \in \mathcal{U}} R_{ij} - \sum_{L_{ij} \in \mathcal{U}} P_{ij}}{\sum_{L_{ij} \in \mathcal{U}} P_{ij}}$$

γ measures the network-wide bandwidth reservation increase caused by the backup LSPs compared to the unprotected case.

Four algorithms for node-failure protection are presented in the following results. The first one (labelled "LOCAL") is as the name suggests a local recovery scheme using only the basic "backup-backup" bandwidth sharing scheme. The second one, "LOCAL with FBW[14]", is an enhanced version taking into account the concept of "primary-backup sharing" (using the vector $F_{ij}(F)$). The same difference exists between the two algorithms labelled respectively "END-TO-END" and "END-TO-END with FBW". It should be noted that the "END-TO-END" algorithm is similar in its behaviour to the algorithm presented in [LWKD02] which we consider to be the state-of-the-art in end-to-end resource sharing.

[14]FBW is an acronym for Freed Bandwidth.

9.7.1.1 Results

Figure 9.13 presents the evolution of γ when we progressively add LSPs to the network. The vertical position of each algorithm is not surprising. However it should be noted how the introduction of vector F improves the performance of the recovery. In terms of resource consumption, local restoration with FBW is as good as end-to-end recovery without FBW. On this topology, the "distance" between the best local approach and the best end-to-end scheme is less than 10%, a price we consider quite cheap to benefit from very short restoration delays.

The decrease of γ that occurs after the establishment of 2000 primary LSPs should be pointed out. This behaviour is due to the method to choose the path of the primary LSP. Indeed, as the primary always follows the minimum hop path, many primary LSPs will tend to overlap while enough bandwidth remains available on this minimum hop path. This tends to create regions where links are used to protect only a small number of nodes. This situation does not create a lot of opportunities to aggregate bandwidth. If we take a look at figure 9.14, we will see that when 2000 primary LSPs and corresponding backups are established in the network, the mean reserved bandwidth is close to 45%. This mean load suggests that some links are completely filled. The following requests thus have to make a detour to avoid the saturated links. Because of this, backup LSPs are now established in other parts of the network where a more important sharing can be realized.

This is confirmed by figure 9.15 which shows the mean number of non-null elements in vector $F_{ij}(F)$, i.e. the evolution of $\frac{1}{|\mathcal{X}|}\sum_{F\in\mathcal{X}} sign(F_{ij}(F))$[15]. Despite not being shown in this work the same kind of behaviour is observed for the density of vector $B_{ij}(F)$. The slope of the curve increases shortly after 2000 primary LSP establishments indicating that backup LSPs are now using links where no bandwidth has been reserved for protecting the same node. This suggests that choosing our primary paths in a smarter way could have a big impact on the amount of sharing.

This simulation proves the interest of including restoration mechanism in the MPLS layer. Indeed lower layer recovery schemes such as SONET self healing rings impose a high level of oversubscription (> 100 %, see [Gro03] for details).

9.7.2 Results on other kinds of topologies

We have used three other topologies. The first two topologies were generated using the BRITE topology generation tool [MLMB01]: one using Waxman's method [Wax88] and the other one using Barabasi-Albert's [BA99]. The third one is the topology of an operational network. The Waxman topology is composed of 50 nodes and 100 full-duplex links. We set the value for parameters α and β to 0.15 and 0.2. The Barabasi-Albert topology is composed of 50 nodes and 97 full-duplex links. The operational network is composed of about 20 nodes and 40 full-duplex links. For the Waxman (WAXMAN) and the Barabasi-Albert (BA) topologies, we have generated an LSP between each pair of nodes. The size of an LSP is chosen according to a uniform random distribution between

[15] $sign(x)$ is equal to 1 if x is positive, to -1 if x is negative and to 0 if x is equal to 0.

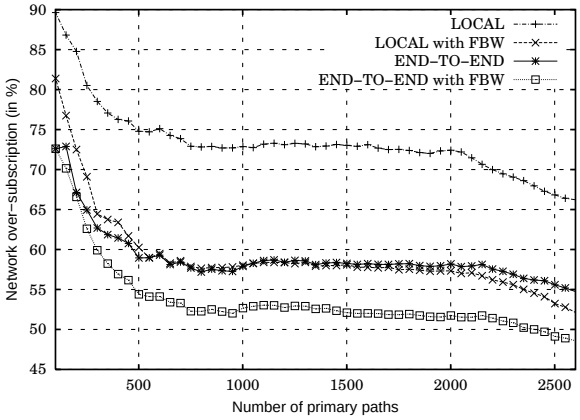

Figure 9.13: Evolution of network oversubscription

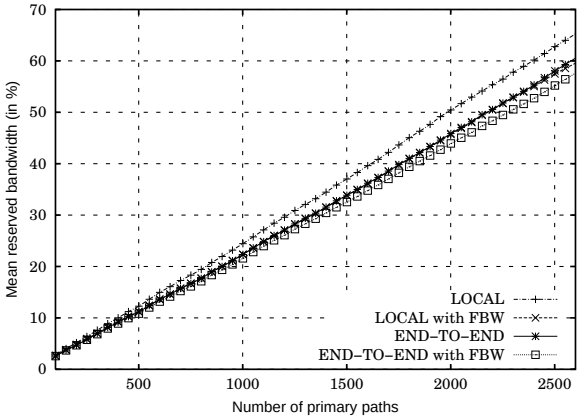

Figure 9.14: Evolution of mean load

5 and 10 units of bandwidth (for comparison, all the links of both topologies have a bandwidth of 1000 units). For the operational network, we have used real traffic measurements (represented in a traffic matrix) to fix the size of the LSP requests. The procedure to obtain the traffic matrix of the operational network is similar to the one described in [BLD+07]. The primary paths were computed with the DAMOTE algorithm [BML03b], which is a good primary path computation algorithm according to simulations in [BLD+07, SBD+05].

Table 9.6 shows the oversubscription values for the three topologies. These values are given after the establishment of all the LSPs leading to a mean reservation of about 40% for WAXMAN and BA topologies and 20% for the operational network topology. We can notice that absolute values for the oversubscription on the WAXMAN and BA topologies are close to the values found in the previous section. On the smaller operational network topology, we notice that the oversubscription level is higher. But even in this case, compared to SONET restoration for which the oversubscription is over 100% in all cases and does not protect against node failures, these results are competitive. Finally, the relative

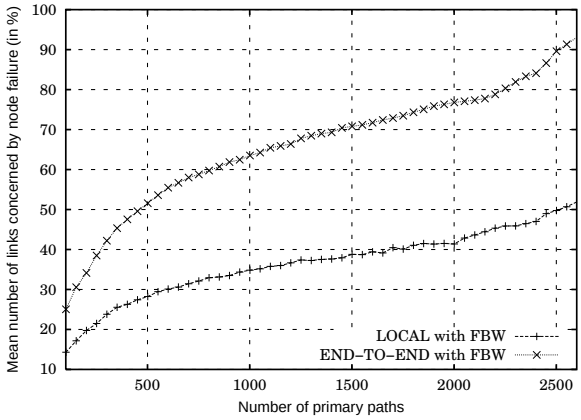

Figure 9.15: Evolution of the number of non-null elements in vector F_{ij}, averaged over all node failures F

	WAXMAN	BA	Operational Network
LOCAL	52.7	65.8	111.4
LOCAL with FBW	50.0	63.2	108.5
E2E	48.6	62.3	93.7
E2E with FBW	43.8	57.4	83.4

Table 9.6: Oversubscription

reduction of bandwidth when primary-backup sharing is used (i.e. with FBW) can go up to 10.3% in the case of end-to-end protection. This means that this kind of sharing is not limited to local restoration and provides good results for end-to-end protection as well.

9.8 Extentions to RSVP-TE and OSPF-TE protocols

We have pointed out that it is possible to extend OSPF-TE and RSVP-TE to support our solution. In this section, we summarize what kind of extensions are needed.

In OSPF-TE, it is necessary to add the primary LSPs reserved bandwidth (P_{ij}).

In RSVP-TE, we have to add:

- In the PATH messages, we have to say if it is a primary or a backup LSP.
  - If it is a primary LSP, we have to add the codepoint which specifies the backup LSP establishment solution. In some cases, we also have to add the $F_{ij}\_objects$.

  - If it is a backup LSP, we have to add the protected link.

- In the RESV messages, we have to add an object containing the computed backup path to be established by the upstream node. Depending on the chosen solution, we can also add an object containing the union of the links used by already computed backup LSPs related to the same primary LSP.

9.9 Conclusion

The contribution of this chapter is threefold. First of all we improved the best known bandwidth sharing scheme without sacrificing simplicity. This new sharing technique is able to provide a substantial decrease of the network oversubscription of both local and end-to-end protection schemes. The second interest of this chapter is to explain the modification required to handle correctly the notion of "preemption levels". The third contribution is to provide a scalable way to implement our efficient bandwidth sharing solution in a distributed way.

Our results show that fast-rerouting is a viable approach to protect traffic that can only accommodate very short interruptions. They also suggest that routing the primary path in a smarter way could help reduce the resource usage further.

10

Conclusions & Perspectives

The Traffic Engineering (TE) research community has been active for several years. Numerous papers have been published on this topic, both from universities and industrial labs, which reflects the huge activity and common interest for this research area.

To solve TE problems researchers use a wide range of mathematical theories and tools, like for example the optimization theory, the control theory, statistics, the art of modeling, distributed systems or game theory. This wide range of potential mathematical theory support contributes to make this research area exciting and interesting. In this work we did use some of these theories while we have improved the way some others are used. We have identified and then solved different TE problems that required innovative solutions.

First we have noted that the mathematical goals of optimization techniques (exact or heuristics) used in TE algorithms were not well defined. The foundation and justification of the choice of the mathematical objective function are often missing or weak, while the quality of that choice is predominant in the overall efficiency of the resulting TE system. From this observation we have performed a deep analysis of the requirements TE algorithms should fulfill and how proposed objective functions reflect such requirements. The results of this study are that all the proposed objective functions are not equivalent, some being far better than others. The delay-based objective functions seem to embed most important TE concerns. This work could be used in future works to revisit main TE algorithms and evaluate the gain that could be obtained by replacing their objective function by a more efficient one. For example we think that all the methods that use the maximal link utilization as only objective and then try to use resulting routing on another traffic matrix[1] are deeply biased. Using a delay-based objective function in this case would greatly improve the obtained results, as this objective function will lead the algorithm to not only reduce the maximal link utilization, but also the utilization of lower loaded links, in addition to improving other TE metrics.

[1] Like for example the model of prediction-based TE which is used in [WXQ+06] to compare to their algorithm.

Then we did tackle a main problem of Link Weight Optimizers (LWO) whose routing models are unaware of the BGP Hot-Potato rule. We have observed that this simplification could lead in the worst case to dramatically bad resulting performance, as shown by simulations based on real data collected on an operational transit network. To solve this problem we have proposed to include interdomain links in the classical LWO problem to be able to model BGP Hot-Potato reroutings as well and to avoid significant errors. The proposed method includes an efficient aggregation technique to keep the problem computationally tractable. We have implemented the proposed LWO, and also tested it on our dataset. Note that including interdomain links in the optimization allowed us to extend the scope of the optimizer to interdomain links in addition to intradomain links. Results were more than satisfactory on this point. An interesting future work could be to extend our work as it was done for non-BGP-aware LWOs to use multiple traffic matrices and also to guarantee good network state in case of network element (link or node) failures. It would also be interesting to develop more specialized objective functions for interdomain links. Typically the cost of using interdomain links is different from the cost of using intradomain links. For example, this cost can be constant until a given load, and proportional to the utilization above that given load. The objective function of such links could reflect this monetary cost to decrease network utilization expenditure. Another interesting future work area is the still open question of optimization oscillations involving neighboring ASes.

The previous study assumed that a particular BGP configuration —an iBGP full-mesh— was used in the network. While this configuration is the BGP default configuration, big networks usually install route reflectors in their topologies, which reduces the number of iBGP sessions to set up in the network, in order to preserve scalability. We have shown that when LWOs are used in a network whose BGP configuration is based on route reflectors, additional problems, like deflections or forwarding loops, could appear *due to the LWO, even if these problems were avoided by design before the optimization.* We have proposed to use a BGP routing solver (C-BGP) inside the LWO to avoid deflections, even in the presence of route reflectors. Again we implemented such LWO and tested it on our real dataset to demonstrate its ability to solve presented problems, but also its efficiency. Note that some recent work presented how to design iBGP topologies based on route reflectors which are not subject to path deflections. One problem of such tools is that the "deflection-free property" is generally only guaranteed for a given set of link weights. If the link weights are changed by a LWO, the property does not hold anymore. We think it would be interesting to combine both approaches, and for example to develop an algorithm able to design an iBGP topology based on route reflectors on which some special LWO can be used without the risk of introducing path deflection. Or we could design an LWO that computes a set of link weights that both balances the load on links and allows an optimal iBGP configuration with a minimal number of iBGP sessions.

A last problem we studied related to LWOs is concerning iBGP multipath load sharing, an optional BGP feature that allows routers to split traffic on multiple equivalent egress routers, like ECMP inside the network. It has been said that this could introduce forwarding loops. We have shown that some general BGP configurations reflecting commercial relationships ensure that forwarding

loops won't be created.

Finally we did study how efficient resilience can be achieved in an MPLS network, in a distributed environment. We focused on fast recovery protection with local backup LSPs. The difficulty of this work was the big amount of information required by computing nodes to perform efficient path computation at reduced bandwidth cost. The simplest solution to flood all the required information in the whole network would not scale. Thus we have proposed a more sophisticated distribution/computation scheme, which allows efficient computation at very reduced bandwidth cost, and which scales in a fully-distributed system.

As a conclusion we have found several innovative solutions to different TE problems. We hope this work has contributed to improve the performance and quality of state of the art Traffic Engineering techniques.

A

Acronyms

ABR — Area Border Router
AS — Autonomous System
BGP — Border Gateway Protocol
CBR — Constraint Based Routing
CSPF — Constrained Shortest Path First
EF — Expedited Forwarding
IGP — Interior Gateway Protocol
IP Prefix — Block of IP Addresses
ISIS — Intermediate System - Intermediate System
ISIS-TE — Traffic Engineering extensions to ISIS
LP — Linear Program
LSA — Link State Advertisement
LSP — Label Switched Path
LSR — Label Switch Router
LWO — Link Weight Optimizer
MPLS — Multi-Protocol Label Switching
OSPF — Open Shortest Path First
OSPF-TE — Traffic Engineering extensions to OSPF
POR — Point Of Repair
PML — Path Merge LSR
QoS — Quality of Service
RSVP — Resource reSerVation Protocol
SLA — Service Level Agreement
SONET — Synchronous Optical NETworking

TE — Traffic Engineering

Bibliography

[ABG+01] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC 3209, Internet Engineering Task Force, December 2001. (Cited on page 3.)

[AC03] David Applegate and Edith Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 313–324. ACM, 2003. (Cited on page 17.)

[ACE+02] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and Principles of Internet Traffic Engineering. RFC 3272, Internet Engineering Task Force, May 2002. (Cited on page 4.)

[ADF+01] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas. LDP Specification. RFC 3036, Internet Engineering Task Force, January 2001. (Cited on page 3.)

[AHX01] D. Awduche, A. Hannan, and X. Xiao. Applicability statement for extensions to RSVP for LSP-tunnels, RFC 3210. Internet Engineering Task Force, December 2001. (Cited on page 138.)

[AJ02] Daniel O. Awduche and Bijan Jabbari. Internet traffic engineering using multi-protocol label switching (MPLS). *Comput. Netw.*, 40(1):111–129, 2002. (Cited on page 4.)

[Ala96] C. Alaettinoglu. Scalable Router Configuration for the Internet. In *Proceedings of IEEE IC3N*, October 1996. (Cited on page 115.)

[AMA+99] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. RFC 2702, Internet Engineering Task Force, September 1999. (Cited on pages 4 and 37.)

[AMO93] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows, theory, algorithms and applications.* Prentice Hall, 1993. (Cited on pages 15 and 20.)

[ANB05] Sharad Agarwal, Antonio Nucci, and Supratik Bhattacharyya. Measuring the Shared Fate of IGP Engineering and Interdomain Traffic. In *ICNP '05: Proceedings of the 13TH IEEE International Conference on Network Protocols*, pages 236–245. IEEE Computer Society, 2005. (Cited on pages 68, 96, 100, and 101.)

[Awd99] D.O. Awduche. MPLS and traffic engineering in IP networks. *Communications Magazine, IEEE*, 37(12):42–47, Dec 1999. (Cited on page 4.)

[BA99] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999. (Cited on page 149.)

[BALM01] W. Ben-Ameur, B. Liau, and N. Michel. Routing strategies for IP networks. *Telektronikk Magazine*, 2/3:145–158, 2/3 2001. (Cited on page 17.)

[BBO+03] Yigal Bejerano, Yuri Breitbart, Ariel Orda, Rajeev Rastogi, and Alexander Sprintson. Algorithms for Computing QoS Paths with Restoration. In *Proc. of IEEE INFOCOM 2003*, volume 22, pages 1435–1445, March 2003. (Cited on page 125.)

[BCC00] T. Bates, R. Chandra, and E. Chen. BGP Route Reflection - An Alternative to Full Mesh IBGP. RFC 2796. April 2000. (Cited on pages 5 and 94.)

[BGP] BGP Best path selection algorithm. http://www.cisco.com/warp/public/459/25.shtml. (Cited on pages 4 and 70.)

[BL06] Simon Balon and Guy Leduc. Dividing the Traffic Matrix to Approach Optimal Traffic Engineering. In *Proceedings of 14th IEEE International Conference on Networks (ICON 2006)*, volume 2, pages 566–571, Singapore, 13-15 Sep. 2006. (Cited on pages 10 and 13.)

[BL07] Simon Balon and Guy Leduc. Can Forwarding Loops Appear When Activating iBGP Multipath Load Sharing? In Springer, editor, *Proceedings of the Third Asian Internet Engineering Conference (AINTEC 2007)*, volume 4866 of *LNCS*, pages 213–225, Phuket, Thailand, Nov. 2007. (Cited on page 13.)

[BL08] Simon Balon and Guy Leduc. Combined Intra- and Inter-domain Traffic Engineering using Hot-Potato Aware Link Weights Optimization. *SIGMETRICS Perform. Eval. Rev.*, 36(1):441–442, 2008. Extended version available on arxiv (arXiv:0803.2824). (Cited on pages 10, 12, 96, 97, and 100.)

[BL09] Simon Balon and Guy Leduc. BGP-aware IGP Link Weight Optimization in Presence of Route Reflectors. *INFOCOM 2009. 28th IEEE International Conference on Computer Communications. Proceedings*, April 20-24 2009. (Cited on page 12.)

[BLD+07] Simon Balon, Jean Lepropre, Olivier Delcourt, F. Skivée, and Guy Leduc. Traffic Engineering an Operational Network with the TOTEM Toolbox. *IEEE Transactions on Network and Service Management*, 4(1):51–61, June 2007. (Cited on pages 12, 13, 23, and 150.)

[BML03a] François Blanchy, L. Mélon, and Guy Leduc. A Preemption-Aware On-line Routing Algorithm for MPLS Networks. *Telecommunication Systems*, 24:187–206, 2-4, Oct.-Dec. 2003. (Cited on pages 19 and 128.)

[BML03b] François Blanchy, Laurent Mélon, and Guy 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. (Cited on pages 42 and 150.)

[BML06] S. Balon, L. Mélon, and G. Leduc. A scalable and decentralized fast-rerouting scheme with efficient bandwidth sharing. *Computer Networks*, 50(16):3043–3063, Nov. 2006. (Cited on pages 12 and 13.)

[BRRT05] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, and M. Thorup. A hybrid genetic algorithm for the weight setting problem in OSPF-IS-IS routing. *Networks*, 46(1):36–56, 2005. (Cited on page 16.)

[BSI02] R. Boutaba, W. Szeto, and Y. Iraqi. DORA: Efficient Routing for MPLS Traffic Engineering. *J. Netw. Syst. Manage.*, 10(3):309–325, 2002. (Cited on page 20.)

[BSL05] Simon Balon, Fabian Skivée, and Guy Leduc. Comparing traffic engineering objective functions. In *CoNEXT 2005, Student Workshop*, Toulouse, France, 24-27 Oct. 2005. (Cited on pages 10 and 13.)

[BSL06] S. Balon, F. Skivée, and G. Leduc. How Well Do Traffic Engineering Objective Functions Meet TE Requirements? In *Proceedings of IFIP Networking 2006, Coim-*

bra, volume 3976, pages 75–86. Springer LNCS, May 2006. (Cited on pages 10 and 13.)

[BUM08] Marc-Olivier Buob, Steve Uhlig, and Mickael Meulle. Designing Optimal iBGP Route-Reflection Topologies. In *Proceedings of IFIP Networking*, volume 4982 of *Lecture Notes in Computer Science*, pages 542–553. Springer, 2008. (Cited on page 95.)

[Cal90] R. Callon. Use of OSI IS-IS for Routing in TCP/IP and Dual Environments. RFC 1195, Internet Engineering Task Force, December 1990. (Cited on page 2.)

[CAR] Cariden MATE. http://www.cariden.com/products/. (Cited on page 24.)

[CEDFQ06] Selin Cerav-Erbas, Olivier Delcourt, Bernard Fortz, and Bruno Quoitin. The Interaction of IGP Weight Optimization with BGP. In *ICISP '06: Proceedings of the International Conference on Internet Surveillance and Protection*, page 9. IEEE Computer Society, 2006. (Cited on pages 66 and 69.)

[CWZ00] Zhiruo Cao, Zheng Wang, and Ellen Zegura. Performance of Hashing-Based Schemes for Internet Load Balancing. In *Proceedings of INFOCOM*, volume 1, pages 332–341, 2000. (Cited on page 2.)

[dBW02] S. Van den Bosh and C. Wittoeck. Traffic Engineering, Global Route Optimisation - a mechanism to increase the traffic-handling capability of existing MPLS/IP networks. Technical report, Alcatel White Paper, 2002. (Cited on page 24.)

[DHdLVPdB03] N. Degrande, G. Van Hoey, P. de La Vallée-Poussin, and S. Van den Busch. Inter-area traffic engineering in a differentiated services network. *J. Networks Syst. Manage.*, 11(4), 2003. (Cited on pages 43 and 45.)

[EJLW01] A. Elwalid, C. Jin, S. H. Low, and Indra Widjaja. MATE: MPLS adaptive traffic engineering. In *Proc. of IEEE INFOCOM*, pages 1300–1309, 2001. (Cited on pages 21 and 43.)

[ERP02] M. Ericsson, Mauricio G. C. Resende, and Panos M. Pardalos. A Genetic Algorithm for the Weight Setting Problem in OSPF Routing. *J. Comb. Optim.*, 6(3):299–333, 2002. (Cited on page 16.)

[Fa02] F. Le Faucheur and al. Multi-Protocol Label Switching (MPLS) Support of Differentiated Services. RFC 3270. Internet Engineering Task Force, May 2002. (Cited on page 131.)

[FB05] Nick Feamster and Hari Balakrishnan. Detecting BGP configuration faults with static analysis. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 43–56, Berkeley, CA, USA, 2005. USENIX Association. (Cited on page 96.)

[FBR03] Nick Feamster, Jay Borkenhagen, and Jennifer Rexford. Guidelines for interdomain traffic engineering. *SIGCOMM Comput. Commun. Rev.*, 33(5):19–30, 2003. (Cited on pages 8 and 73.)

[FBR04] Nick Feamster, Hari Balakrishnan, and Jennifer Rexford. Some Foundational Problems in Interdomain Routing. In *3rd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, San Diego, CA, November 2004. (Cited on page 95.)

[FGL+00] Anja Feldmann, Albert Greenberg, Carsten Lund, Nick Reingold, and Jennifer Rexford. Netscope: Traffic engineering for IP networks. *IEEE Network Magazine*, 14(2):pp. 11–19, March/April 2000. (Cited on pages 24 and 68.)

[FGL+01] Anja Feldmann, Albert Greenberg, Carsten Lund, Nick Reingold, Jennifer Rexford, and Fred True. Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE/ACM Transactions on Networking*, pages 265–279, June 2001. (Cited on pages 26 and 68.)

[FKF06] Simon Fischer, Nils Kammenhuber, and Anja Feldmann. REPLEX: dynamic traffic engineering based on wardrop routing policies. In *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*, pages 1–12. ACM, 2006. (Cited on page 21.)

[Fou] Foundry enterprise configuration and management guide. http://www.foundrynet.com/services/documentation/ecmg/BGP4.html#17143. (Cited on pages 4 and 70.)

[FRT02] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *Communications Magazine, IEEE*, 40(10):118–124, Oct 2002. (Cited on pages 7 and 16.)

[FT00] B. Fortz and M. Thorup. Internet Traffic Enginnering by Optimizing OSPF Weights. In *Proc. IEEE INFOCOM 2000*, pages 519–528, 2000. (Cited on pages 7, 16, 40, 45, 54, and 68.)

[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. (Cited on pages 16 and 68.)

[FT03] B. Fortz and M. Thorup. Robust optimization of OSPF/IS-IS weights. In *Proceedings of INOC*, pages 225–230, October 2003. (Cited on pages 16 and 68.)

[FT04] Bernard Fortz and Mikkel Thorup. Increasing Internet Capacity Using Local Search. *Comput. Optim. Appl.*, 29(1):13–48, 2004. (Cited on pages 16 and 106.)

[Gal77] R. Gallager. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communications*, 25(1):73–85, Jan 1977. (Cited on page 21.)

[GDZ05] Ruomei Gao, Constantinos Dovrolis, and Ellen W. Zegura. Interdomain Ingress Traffic Engineering Through Optimized AS-Path Prepending. In *Proceedings of IFIP Networking 2005*, volume 3462 of *Lecture Notes in Computer Science*, pages 647–658, Waterloo, Canada, 2005. Springer. (Cited on page 89.)

[GOW97] R.A. Guerin, A. Orda, and D. Williams. QoS routing mechanisms and OSPF extensions. *Global Telecommunications Conference, 1997. GLOBECOM '97., IEEE*, 3:1903–1908 vol.3, 3-8 Nov 1997. (Cited on page 19.)

[GR01] Lixin Gao and Jennifer Rexford. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking*, pages 681–692, December 2001. (Cited on page 115.)

[Gro03] Wayne D. Grover. *Mesh-Based Survivable Networks : Options and Strategies for Optical, MPLS, SONET, and ATM Networking*. Prentice Hall, 2003. (Cited on page 149.)

[GS98] Wayne D. Grover and Demetrios Stamatelakis. Cycle-Oriented Distributed Preconfiguration: Ring-like Speed with Mesh-like Capacity for Self-planning Network Restoration. In *Proc. of IEEE ICC 1998*, pages 537–543, Atlanta, June 1998. (Cited on page 125.)

[GW02] Timothy G. Griffin and Gordon Wilfong. On the correctness of IBGP configuration. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 17–29. ACM, 2002. (Cited on pages 94, 95, 99, and 100.)

[HBJ06] Joel M. Halpern, Manav Bhatia, and Paul Jamka. Advertising Equal Cost Multipath Routes in BGP. *Internet Draft, Work In Progress*, February 2006. (Cited on page 114.)

[HR08] Jiayue He and J. Rexford. Toward internet-wide multipath routing. *Network, IEEE*, 22(2):16–21, March-April 2008. (Cited on page 113.)

[HRC07] Jiayue He, Jennifer Rexford, and Mung Chiang. Don't optimize existing protocols, design optimizable protocols. *SIGCOMM Comput. Commun. Rev.*, 37(3):53–58, 2007. (Cited on page 18.)

[IB02] I. Iliadis and D. Bauer. A New Class of Online Minimum-Interference Routing Algorithms. In LNCS, editor, *Proceedings of IFIP Networking 2002*, volume 2345, pages 959–971, 2002. (Cited on page 20.)

[ICBD04] Gianluca Iannaccone, Chen-Nee Chuah, Supratik Bhattacharyya, and Christophe Diot. Feasibility of IP restoration in a tier 1 backbone. *IEEE Network*, 18(2), 2004. (Cited on pages 76 and 113.)

[JAC+02] B. Jamoussi, L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, E. Gray, J. Heinanen, T. Kilty, and A. Malis. Constraint-Based LSP Setup using LDP. RFC 3212, Internet Engineering Task Force, January 2002. (Cited on page 3.)

[KGST07] Kin Wah Kwong, Roch Guerin, Anees Shaikh, and Shu Tao. Improving service differentiation in IP networks through dual topology routing. In *CoNEXT '07: Proceedings of the 2007 ACM CoNEXT conference*, pages 1–12, New York, NY, USA, 2007. ACM. (Cited on page 62.)

[KKDC05] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. Walking the Tightrope: Responsive Yet Stable Traffic Engineering. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 253–264, New York, NY, USA, 2005. ACM. (Cited on page 21.)

[KKL00] K. Kar, M. Kodialam, and T.V. Lakshman. Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. *Selected Areas in Communications, IEEE Journal on*, 18(12):2566–2579, Dec 2000. (Cited on page 20.)

[KKL02] Koushik Kar, Murali Kodialam, and T.V. Lakshman. Routing Restorable Bandwidth Guaranteed Connections using Maximum 2-Route Flows. In *Proc. of IEEE INFOCOM 2002*, volume 21, pages 113–121, 2002. (Cited on page 124.)

[KKY03] D. Katz, K. Kompella, and D. Yeung. Traffic Engineering (TE) Extensions to OSPF Version 2. RFC 3630. Internet Engineering Task Force, September 2003. (Cited on pages 19 and 139.)

[KL00] Murali S. Kodialam and T. V. Lakshman. Minimum Interference Routing with Applications to MPLS Traffic Engineering. In *Proc. of IEEE INFOCOM 2000*, pages 884–893, 2000. (Cited on pages 41 and 124.)

[KL01] Murali Kodialam and T.V. Lakshman. Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels using Aggregated Link Usage Information. In *Proc. of IEEE INFOCOM 2001*, pages 376 – 385, April 2001. (Cited on pages 125 and 139.)

[KL03] Murali Kodialam and T.V. Lakshman. Dynamic Routing of Restorable Bandwidth-Guaranteed Tunnels Using Aggregated Network Resource Usage Information. *IEEE/ACM Transactions on Networking*, 11(3):399 – 410, June 2003. (Cited on page 125.)

[KLOS06] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta. A Versatile Scheme for Routing Highly Variable Traffic in Service Overlays and IP Backbones. *INFOCOM 2006. 25th IEEE*

International Conference on Computer Communications. Proceedings, pages 1–12, April 2006. (Cited on page 17.)

[KLS06] M. Kodialam, T. Lakshman, and Sudipta Sengupta. Throughput Guaranteed Restorable Routing Without Traffic Prediction. In *ICNP '06: Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 137–146. IEEE Computer Society, 2006. (Cited on page 17.)

[LAB+06] Guy Leduc, H. Abrahamsson, Simon Balon, S. Bessler, M. D'Arienzo, Olivier Delcourt, J. Domingo-Pascual, S. Cerav-Erbas, I. Gojmerac, X. Masip, A. Pescaph, B. Quoitin, S.F. Romano, E. Salvatori, F. Skivée, H.T. Tran, S. Uhlig, and H. Ümit. An open source traffic engineering toolbox. *Computer Communications*, 29(5):593–610, March 2006. (Cited on pages 13, 23, and 45.)

[LL05] F. Le Faucheur and W. Lai. Maximum Allocation Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering. Internet Engineering Task Force, RFC 4125, June 2005. (Cited on page 20.)

[LM04] Youngseok Lee and Biswanath Mukherjee. Traffic engineering in next-generation optical networks. *IEEE Communications Surveys & Tutorials*, 6(3):16–33, 2004. (Cited on page 15.)

[LSV04] Jean-François Lalande, Michel Syska, and Yann Verhoeven. Mascopt - A Network Optimization Library: Graph Manipulation. Technical report, Sophia Antipolis, Projet Mascotte, Avril 2004. (Cited on page 24.)

[LT01] Yu Liu and David Tipper. Spare capacity allocation for nonlinear cost and failure-dependent path restoration. In *Proc. of the Third International Workshop on Design of Reliable Communication Networks, DRCN 2001*, Budapest, Hungary, October 7–10 2001. (Cited on page 148.)

[LTS01] Yu Liu, David Tipper, and Peerapon Siripongwutikorn. Approximating optimal spare capacity allocation by successive survivable routing. In *Proc. of IEEE INFOCOM 2001*, pages 699–708, April 2001. (Cited on page 148.)

[LWKD02] Guangzhi Li, Dongmei Wang, Charles Kalmanek, and Robert Doverspike. Efficient Distributed Path Selection for Shared Restoration Connections. In *Proc. of IEEE INFOCOM 2002*, volume 21, pages 140 – 149, June 2002. (Cited on pages 125 and 148.)

[LYDW01] Guangzhi Li, Jennifer Yates, Robert Doverspike, and Dongmei Wang. Experiments in Fast Restoration using GMPLS in Optical / Electronic Mesh Networks. In *Optical Fiber Commun. Conf.*, March 2001. (Cited on page 124.)

[MBL03] Laurent Mélon, François Blanchy, and Guy Leduc. Decentralized local backup LSP calculation with efficient bandwidth sharing. In *Proc. of 10th International Conference on Telecommunications (ICT'2003)*, pages 929–937, Papeete, Tahiti, 23-28 Feb. 2003. IEEE Press. (Cited on pages 11, 126, and 127.)

[MCSA03] Jose L. Marzo, Eusebi Calle, Caterina Scoglio, and Tricha Anjali. QoS Online Routing and MPLS Multilevel Protection: A Survey. *Communications Magazine, IEEE*, 41(10):126–132, Oct 2003. (Cited on pages 8 and 15.)

[MIB+04] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, and Christophe Diot. Characterization of Failures in an IP Backbone Network. In *Proc. of IEEE INFOCOM 2004*, volume 23, pages 2307–2317, Hong Kong, March 2004. (Cited on page 124.)

[MK04a] E. Mulyana and U. Killat. Impact of partial demand increase on the performance of IP networks and re-optimization approaches. In *Proceedings of the 3rd Polish-German Teletraffic Symposium PGTS*, pages 275–284, Dresden Germany, September 2004. (Cited on page 17.)

[MK04b] E. Mulyana and U. Killat. Optimization of IP networks in various hybrid IGP/MPLS routing schemes. In *Proceedings of the 3rd Polish-German Teletraffic Symposium PGTS*, pages 295–304, Dresden Germany, September 2004. (Cited on page 17.)

[MLMB01] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. BRITE: An approach to universal topology generation. In *In Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS '01*, Cincinnati, Ohio, Août 2001. (Cited on pages 61 and 149.)

[MM05] M. Menth and R. Martin. Network resilience through multi-topology routing. *Design of Reliable Communication Networks, 2005. (DRCN 2005). Proceedings.5th International Workshop on*, pages 271–277, Oct. 2005. (Cited on page 62.)

[Moy98] J. Moy. OSPF Version 2. RFC 2328. April 1998. (Cited on pages 2 and 66.)

[MTS+02] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: existing techniques and new directions. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–174. ACM, 2002. (Cited on page 26.)

[MU03] E. Mulyana and Killat. U. An offline hybrid IGP/MPLS traffic engineering approach under LSP constraints. In *Proceedings of the 1st International Network Optimization Conference INOC 2003*, Evry/Paris France, oct 2003. (Cited on page 44.)

[MWA05] Ratul Mahajan, David Wetherall, and Thomas Anderson. Negotiation-Based Routing Between Neighboring ISPs. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 29–42. USENIX Association, 2005. (Cited on pages 89 and 90.)

[NSB+03] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, and C. Diot. IGP Link Weight Assignment for Transient Link Failures. In *Proceedings of 18th International Teletraffic Congress (ITC)*, September 2003. (Cited on page 16.)

[OPN] Opnet products. http://www.opnet.com. (Cited on page 24.)

[PMR+07] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault. Multi-Topology (MT) Routing in OSPF. RFC 4915, Internet Engineering Task Force, June 2007. (Cited on pages 53 and 62.)

[Pre] B. J. Premore. *SSF Implementations of BGP-4*. (Cited on page 18.)

[PSS08] Tony Przygienda, Naiming Shen, and Nischal Sheth. M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs). RFC 5120, Internet Engineering Task Force, February 2008. (Cited on pages 53 and 62.)

[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)*, pages 450–457, Rome, Italy, April 18-20th 2005. (Cited on page 90.)

[QPBU05] B. Quoitin, C. Pelsser, O. Bonaventure, and S. Uhlig. A performance evaluation of BGP-based traffic engineering. *International Journal of Network Management*, 15(3):177–191, 2005. (Cited on page 18.)

[QU05] B. Quoitin and S. Uhlig. Modeling the routing of an Autonomous System with C-BGP. *IEEE Network*, 19(6):12–19, November 2005. (Cited on pages 101 and 105.)

[QUP+03] B. Quoitin, S. Uhlig, C. Pelsser, L. Swinnen, and O. Bonaventure. Interdomain traffic engineering with BGP. *Communications Magazine, IEEE*, 41(5):122–128, May 2003. (Cited on pages 8 and 89.)

[QX02] Chunming Qiao and Dahai Xu. Distributed Partial Information Management (DPIM) Schemes for Survivable Networks - Part I. In *Proc. of IEEE INFOCOM 2002*, volume 21, pages 302 – 311, June 2002. (Cited on page 125.)

[Rex06] Jennifer Rexford. *Handbook of Optimization in Telecommunications*, chapter Route optimization in IP networks. Springer Science + Business Media, February 2006. (Cited on pages 15, 68, 96, and 100.)

[Rie03] A. Riedl. Optimized routing adaptation in IP networks utilizing OSPF and MPLS. In *Proceedings of IEEE ICC*, volume 3, pages 1754–1758, May 2003. (Cited on page 17.)

[RTZ03] Matthew Roughan, Mikkel Thorup, and Yin Zhang. Traffic Engineering with Estimated Traffic Matrices. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 248–258. ACM, 2003. (Cited on page 69.)

[SAC+04] C. Scoglio, T. Anjali, J. Cavalcante, I. Akyildiz, and G. Uhl. TEAM: A Traffic Engineering Automated Manager for DiffServ-Based MPLS Networks. *IEEE Communications Magazine*, 42(10):134–145, 2004. (Cited on page 24.)

[SBD+05] Fabian Skivée, Simon Balon, Olivier Delcourt, Jean Lepropre, and Guy Leduc. Architecture d'une boîte à outils d'algorithmes d'ingénierie de trafic et application au réseau GÉANT. In Richard Castanet, editor, *Actes de Colloque Francophone sur l'Ingénierie des Protocoles (CFIP)*, volume Ingénierie des protocoles - Qualité de service, multimédia et mobilité, pages 317–332, Bordeaux, France„ 29 Mar.-1 Avr. 2005. Hermès Lavoisier. (Cited on pages 12, 14, and 150.)

[SBL06] Fabian Skivée, Simon Balon, and Guy Leduc. A scalable heuristic for hybrid IGP/MPLS traffic engineering - Case study on an operational network. In *Proceedings of 14th IEEE International Conference on Networks (ICON 2006)*, volume 2, pages 572–577, Singapore, 13-15 Sep. 2006. (Cited on pages 8, 13, and 16.)

[SG00] Demetrios Stamatelakis and Wayne D. Grover. IP Layer Restoration and Network Planning Based on Virtual Protection Cycles. *IEEE Journal on selected areas in communications*, 18(10), 2000. (Cited on page 125.)

[SG05] Ashwin Sridharan and Roch Guerin. Making IGP Routing Robust to Link Failures. In *Proceedings of IFIP Networking 2005*, volume 3462 of *Lecture Notes in Computer Science*, pages 634–646, Waterloo, Canada, 2005. Springer. (Cited on page 16.)

[SGD05] Ashwin Sridharan, Roch Guérin, and Christophe Diot. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. *IEEE/ACM Trans. Netw.*, 13(2):234–247, 2005. (Cited on page 18.)

[SKK04] Shan Sinha, Srikanth Kandula, and Dina Katabi. Harnessing TCPs Burstiness using Flowlet Switching. In *3rd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, San Diego, CA, November 2004. (Cited on page 3.)

[SMD03] Ashwin Sridharan, Sue B. Moon, and Christophe 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. ACM Press, 2003. (Cited on page 76.)

[SMH03] V. Sharma, Metanoia, and F. Hellstrand. Framework for Multi-Protocol Label Switching (MPLS)-based Recovery. RFC 3469. Internet Engineering Task Force, 2003. (Cited on pages 124 and 129.)

[SMW02] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with rocketfuel. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 133–145. ACM, 2002. (Cited on page 17.)

[Sob05] J. L. Sobrinho. An Algebraic Theory of Dynamic Network Routing. *IEEE/ACM Trans. Netw.*, 13(5):1160–1173, 2005. (Cited on page 115.)

[SS04] N. Shen and H. Smith. Calculating Interior Gateway Protocol (IGP) routes over traffic engineering tunnels. RFC 3906, Internet Engineering Task Force, October 2004. (Cited on page 17.)

[Ste99] J. Stewart. *BGP4 : Interdomain routing in the Internet*. Addison Wesley, 1999. (Cited on pages 4, 114, and 116.)

[SWBW03] Subhash Suri, Marcel Waldvogel, Daniel Bauer, and Priyank Ramesh Warkhede. Profile-based routing and traffic engineering. *Computer Communications*, 26(4):351–365, March 2003. (Cited on page 42.)

[TDRR05] Renata Teixeira, Nick Duffield, Jennifer Rexford, and Matt Roughan. Traffic matrix reloaded: Impact of routing changes. In *Proceedings of Passive and Active Measurement*, volume 3431 of *LNCS*, pages 251–264. Springer, March/April 2005. (Cited on page 69.)

[TOT] http://totem.run.montefiore.ulg.ac.be.
(Cited on pages 23, 61, and 63.)

[TSGR04] Renata Teixeira, Aman Shaikh, Tim Griffin, and Jennifer Rexford. Dynamics of hot-potato routing in IP networks. volume 32, pages 307–319, New York, NY, USA, 2004. ACM. (Cited on page 69.)

[Tya02] H. Tyan. *Design, realization and evaluation of a component-based compositional software architecture for network simulation*. Ohio State University, 2002. (Cited on page 18.)

[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)*, pages 75–82, Rome, Italy, April 18-20th 2005. (Cited on page 8.)

[UQLB06] S. Uhlig, B. Quoitin, Jean Lepropre, and Simon Balon. Providing public intradomain traffic matrices to the research community. *SIGCOMM Comput. Commun. Rev.*, 36(1):83–86, 2006. (Cited on pages 12 and 13.)

[Van05] Virginie Van den Schrieck. Conception d'un langage flexible de définition de politiques de routage BGP. Master's thesis, Université Catholique de Louvain (UCL), Belgium, June 2005. (Cited on page 101.)

[Vas79] Kenneth Vastola. A Numerical Study of Two Measures of Delay for Network Routing. Master's thesis, Illinois University at Urbana-Champaign, Coordinated Science Lab, Sep. 1979. (Cited on page 36.)

[VVKB06] Mythili Vutukuru, Paul Valiant, Swastik Kopparty, and Hari Balakrishnan. How to Construct a Correct and Scalable iBGP Configuration. In *Proceedings of INFOCOM*, pages 1–12, Barcelona, Spain, April 2006. (Cited on pages 95, 96, 99, and 100.)

[W.59] Dijkstra E. W. A note on two problems in connection with graphs. Numerische Mathematik, 1959. (Cited on page 133.)

[WAN] IP/MPLSView (Wandl). http://www.wandl.com/html/mplsview/. (Cited on page 24.)

[Wax88] B.M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1671–1622, December 1988. (Cited on pages 45 and 149.)

[WHPH08] N. Wang, K.H. Ho, G. Pavlou, and M. Howarth. An overview of routing optimization for internet traffic engineering. *Communications Surveys & Tutorials, IEEE*, 10(1):36–56, First Quarter 2008. (Cited on page 15.)

[WXQ+06] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. COPE: Traffic Engineering in Dynamic Networks. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 99–110. ACM, 2006. (Cited on pages 17, 69, and 153.)

[WYXN05] Jun Wang, Yaling Yang, Li Xiao, and Klara Nahrstedt. Edge-based traffic engineering for OSPF networks. *Computer Networks*, 48(4):605–625, July 2005. (Cited on page 41.)

[XCR07] Dahai Xu, Mung Chiang, and Jennifer Rexford. DEFT: Distributed Exponentially-weighted Flow Splitting. In *Proceedings of IEEE INFOCOM, International Conference on Computer Communications*, pages 71–79, May 2007. (Cited on page 18.)

[XCR08] Dahai Xu, Mung Chiang, and Jennifer Rexford. Link-State Routing with Hop-by-Hop Forwarding Can Achieve Optimal Traffic Engineering. In *Proceedings of IEEE INFOCOM, International Conference on Computer Communications*, pages 466–474, April 2008. (Cited on page 18.)

[XHBN00] Xipeng Xiao, A. Hannan, B. Bailey, and L.M. Ni. Traffic engineering with MPLS in the Internet. *Network, IEEE*, 14(2):28–33, Mar/Apr 2000. (Cited on page 4.)

[YF03] Ossama Younis and Sonia Fahmy. Constraint-Based Routing in the Internet: Basic Principles and Recent Research. *IEEE Communications Surveys & Tutorials*, 5(1), 2003. (Cited on page 15.)

[Yua03] Di Yuan. A bicriteria optimization approach for robust OSPF routing. *IP Operations and Management, 2003. (IPOM 2003). 3rd IEEE Workshop on*, pages 91–98, 1-3 Oct. 2003. (Cited on page 16.)

[YXW+05] Y.R. Yang, Haiyong Xie, Hao Wang, A. Silberschatz, A. Krishnamurthy, Liu Yanbin, and Li Erran Li. On route selection for interdomain traffic engineering. *IEEE Network*, 19(6):20–27, Nov.-Dec. 2005. (Cited on page 90.)

[ZRDG03] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 206–217, San Diego, CA, USA, 2003. ACM. (Cited on page 26.)