Traffic Engineering an Operational Network with the TOTEM Toolbox

Simon Balon, Jean Lepropre, Olivier Delcourt, Fabian Skivée, and Guy Leduc, *Member, IEEE*

***Abstract*—We explain how the TOTEM toolbox can be used to engineer an operational network. TOTEM is an open source TOolbox for Traffic Engineering Methods which covers IP-based and MPLS-based intradomain traffic engineering (TE) algorithms, but also interdomain TE. In this paper, we use the toolbox as an off-line simulator to optimise the traffic of an operational network. To help an operator to choose between an IP-based or MPLS-based solution, or to find the best way to load-balance a network for a given traffic, our case study compares several IP and MPLS routing algorithms, evaluates the impact of hot-potato routing on the intradomain traffic matrix, and analyses the worst-case link failure. This study reveals the power of a toolbox that federates many traffic engineering algorithms.**

***Index Terms*—TOTEM, Traffic Engineering, TE, MPLS, IP, BGP**

I. Introduction

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. The main objective of the TOTEM toolbox ([1], [2]) is to reconcile the academic and the operational worlds by providing inter-operable 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 utilisation modes. It can be deployed either as an on-line tool in an operational network or as an off-line traffic engineering simulator. Moreover, a large variety of traffic engineering methods are integrated. These methods can be classified with respect to different axes like intradomain or interdomain, on-line or off-line, IP or MPLS (Multi Protocol Label Switching), centralised or distributed.

TOTEM is a useful tool for network operators. With so many algorithms combined in a common framework it is possible to test and evaluate several engineering solutions quickly. The TOTEM toolbox is described in [2], which presents the toolbox, its architecture, and a series of algorithms/methods that are (or could be) integrated in it. The present paper applies the toolbox to a real case study. It illustrates the utility of our toolbox for a network operator and shows how different TE algorithms can be combined for a particular purpose (i.e. to see the effects of hot-potato routing in this case), which is one of the key advantages of a toolbox federating those algorithms. The utility of the toolbox is much higher than the sum of the utilities of the embedded algorithms. This case study has been performed on an operational multi-gigabit network whose topology is composed of about 20 nodes and 40 links. Link capacities go from 155Mbps to 10Gbps (about 43% are 10 Gbps links, 3% are 5 Gbps and 54% are $\leq$ 2.5 Gbps).

The goal of the case study is to provide a set of answers to a wide variety of questions a network operator may have. For example, to the question “Is it worthwhile to deploy MPLS in my network?”, we provide the best routing schemes available in IP and MPLS networks. With such a study in hand, an operator can choose one among all possible solutions with full knowledge of their pros and cons. To the question “Which link is the most critical in my network?”, we provide a study of worst case link failure. We also describe how to infer traffic matrices on the network, considering hot-potato effects. This is useful for an operator which needs some information about the impact of a link metric change on the traffic of his/her network. Note that it is also possible to evaluate the network protection cost using local or global backup LSPs with the TOTEM toolbox. Such a study can be found in [3].

The paper is structured as follows. In section II, we introduce traffic engineering concepts. Section III briefly describes the TOTEM toolbox and section IV details algorithms used in the case study. Section V presents how to compute the traffic matrix from netflow traces and the influence of link metrics on the traffic matrix via Hot Potato routing effects. In section VI we analyse the traffic and the network state considering all possible IP and MPLS algorithms while section VII analyses the effects of link failures on the network state. Finally, section VIII concludes this study.

II. Traffic engineering

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. But how can this be achieved in practice?

Consider a network running a classical intradomain IGP[1] 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

Manuscript received March 21, 2006; revised Augustus 10, 2006 and December 15, 2006; Accepted for publication January 29, 2007. Editor : George Pavlou, University of Surrey, UK.

The authors are with Research Unit in Networking (RUN), Université de Liège (ULg), Belgium (e-mail:Guy.Leduc@ulg.ac.be).

S. Balon and J. Lepropre are Research Fellows of the Belgian National Fund for the Scientific Research (F.N.R.S).

[1]Interior Gateway Protocol

minimum hop routing) or to the inverse of the capacity of the links (as recommended by CISCO), for example.

This (simple) routing configuration does 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 at least to a high delay due to packet queueing. 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 free 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).

The first technique that can achieve such an objective 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 minimised. This technique requires to know some information about the network traffic, which is usually aggregated and represented as a traffic matrix. The problem of finding the set of weights that minimises the load of the most loaded link(s) is combinatorial and some heuristics to solve it have been proposed in [4].

A shortcoming of this approach is that these weights are optimised for a given traffic matrix. If the actual traffic is different, the optimum is not reached anymore. If we compute a new set of link weights and update them in the network, this may also lead to transient routing loops, if the updates are not done in the right order. To circumvent this, a possible solution is to optimise the set of weights for several traffic matrices, such that the routing conditions are *quite good* for all of them, without being really optimal for any.

A similar problem occurs when a link or a node fails. After a transient period, the flows that were previously routed over the failed resource are now routed on other links, which can also lead to congested links. A solution to this problem can be to optimise the set of weights so that the load is still reasonably balanced under any failure scenario.

We notice that the problem gets more and more complex, while adding new objectives (several traffic matrices, several failure scenarios, ...). A combination of these objectives should also reflect that the network should be better optimized under normal conditions than under failure. Basically, if there are m links in the topology, the optimizer has only $m - 1$ variables to tune to find the best compromise.

A completely different solution to this complex problem is to factorise it into several simpler ones. MPLS is a technology that allows one to establish tunnels, called Label Switched Paths (LSPs), in the network. Thanks to these LSPs, 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 optimising 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 [5]. 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. This is true even though IP routing can be optimized for a given objective function and a given traffic matrix. 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.

III. TOTEM TOOLBOX

In this section we briefly present the TOTEM toolbox. More information about online and offline deployment of the toolbox or its architecture can be found in [2].

A. *Toolbox-related work*

Several network optimisation tools exist, e.g., MATE (Cariden), Netscope (AT&T), Tunnel Builder Pro (CISCO), TSOM (Alcatel), Conscious (Zvolve), IP/MPLSView (Wandl) and SP Guru (Opnet). All these tools are centralised and propose exact and heuristic optimisation methods. We refer to [2] for a complete review of toolbox-related works.

TOTEM is different from all other network optimisation 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 optimisation, primary and backup LSP routing, and BGP[2] simulations. These methods can be easily compared, combined and extended.

B. *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 utilisation);
- **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;
- **Scenarios module:** contains the classes related to simulation scenarios providing the ability to read, execute or generate scenarios (explained below);
- **Algorithm repository:** contains all the traffic engineering algorithms. This is the central part of the toolbox. The algorithms of this repository that are used in this study are described in detail in section IV;
- **Chart module:** contains some functionalities related to charts generation. This module allows the user to automatically generate charts using various data sources;

[2]Border Gateway Protocol

- **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.

C. *Simulation scenarios*

To simplify the use of the toolbox in simulation mode, we set up a kind of scripting language by means of scenario XML files. The content of a scenario XML 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 already allow to build very 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;
- optimise the IGP weights using IGP-WO[3];
- display the link loads with updated weights.

All the results presented in section V were obtained thanks to the toolbox and scenario files.

The language defined by the scenario XML 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 into the toolbox during runtime.

D. *Data flows in the toolbox*

The process to engineer a network from data collection to analysis report is described in Figure 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 III-C) that will control the toolbox execution. The toolbox will simulate the scenario and produce some reports (text file or simple graph). With this process, it is simple to simulate link failure, traffic matrix evolution or IGP metric optimisation and to analyse the impact on link loads, path delay variation or other kind of operational requirements. It is also possible to replace the simulation scenario with the use of the Graphical User Interface.

IV. TE ALGORITHMS USED IN THIS PAPER

In this section, we present the algorithms integrated in the toolbox that we will use in our simulations.

A. *Classical algorithms*

We first describe basic algorithms, which can be used as a starting point for comparison purposes or as building blocks of more complex methods.

One well-known problem is to find the shortest path, the minimum cost path or the minimum total weight path between two nodes of a network. We have implemented several algorithms that perform shortest path computations (more information in [6], [7]), including:

- Dijkstra’s shortest path algorithm (SPF): it computes the shortest path tree to all other nodes, or simply a path to a given node. We have used a priority queue under the form of a binary heap to implement this algorithm. This allows us to obtain a complexity of $\mathcal{O}((V+E)\log V)$ where V is the number of vertices (nodes) and E is the number of edges (links);
- CSPF (Constraint Shortest Path First): it computes the shortest path 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 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.

B. *IGP-WO*

The IGP-WO [4] (*Interior Gateway Protocol-Weight Optimisation*) module aims at finding a link weights setting in the domain for an optimal load balancing. It provides a routing scheme adapted to one or more traffic matrices.

The main inputs to the IGP-WO module are:

- Network topology: routers (nodes), links (arcs) and link capacities;
- Traffic matrices: for each (origin, destination) pair, the requested bandwidth.

The program provides as output a set of weights for the links in the network. If multiple paths of equal cost exist in the network, the traffic is supposed to be split equally among all of these shortest paths. This is known as equal cost multipath (ECMP). The algorithm tries to minimise an objective function which is explained in section VI-A.

Since the problem of finding the optimal weight setting is NP-hard (no efficient algorithm available), a heuristic algorithm is applied to find a *good but not necessarily optimal* solution. The algorithm is based on a well-known metaheuristic technique, called tabu search.

C. *DAMOTE*

DAMOTE [8] (Decentralised Agent for MPLS Online Traffic Engineering) is a routing algorithm whose purpose is to compute LSPs under constraint. DAMOTE is more sophisticated than a mere CSPF algorithm. The difference is that DAMOTE finds the path that minimises a given objective function under bandwidth constraints. Examples of such objective functions are: resource utilisation (DAMOTE operates as a CSPF with a hop-count metric in this case), load balancing, hybrid load balancing (where long detours are penalised), preemption-aware routing (where induced reroutings are penalised).

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, we can define a capacity constraint for different class types (CT) of traffic.

[3]IGP-WO (*Interior Gateway Protocol-Weight Optimiser*) is described in section IV-B.

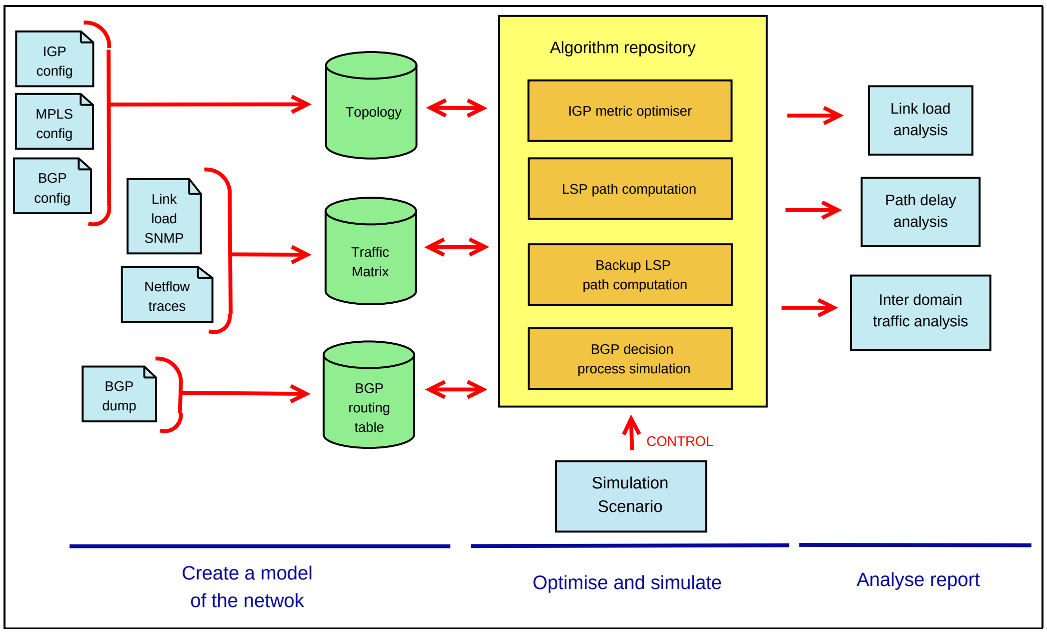

Fig. 1. *Traffic engineering analysis using the toolbox*

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 only the reserved bandwidth of lower preemption level LSPs (these LSPs are more important). This allows us 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 [9] (Maximum Allocation Model) which has been proposed by the IETF MPLS Diff-Serv working group.

DAMOTE can also compute backup LSPs ([3]). Each primary LSP is protected by a series of local detour LSPs. These backup LSPs start from the immediately upstream node of each link of the primary LSP. Doing so, they protect the downstream node (if possible) and the downstream link. They merge with the primary LSP somewhere between the protected resource (exclusive) and the egress node (inclusive). These LSPs have to be established in advance if rapid restoration is required. Also, bandwidth must be reserved for these LSPs, as we want to be sure that there will be enough free bandwidth on all the links in case of failure. In terms of bandwidth consumption, this scheme is only efficient if detour LSPs can share bandwidth among them or with primary LSPs. DAMOTE achieves this under the hypothesis that only one node or link will fail at the same time (*single failure hypothesis*)[4].

DAMOTE can achieve a full protection of all the primary LSPs against link and node failures for an increase in the bandwidth consumption of less than 100% of the resources reserved for primary LSPs (depending on the topology). For comparison, SDH/SONET-based protection leads to an increase of bandwidth consumption of 100% without protecting the nodes of the network.

[4]This assumption is implicit in all the single backup path-based 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.

D. C-BGP

C-BGP is a BGP routing solver ([10]). It aims at computing the interdomain routes selected by BGP routers in a domain. The route computation 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. First, the knowledge of the interdomain routes learned through BGP from the neighbor domains is required. This information can be obtained from MRT dumps collected on the genuine BGP routers or it can be introduced manually. The route computation also relies on the knowledge of the intradomain structure.

The internal representation of the domain also contains a model of BGP routing. This includes nodes that are modelled as BGP routers and the BGP sessions that are established between the BGP routers.

Then, C-BGP makes it possible to run the path computation and later extract information on the path computation results.

V. Measuring the Traffic Matrix and the Effects of Hot Potato Routing

A. Traffic matrix computation

To execute our simulations, we have produced a high number of traffic matrices (TM) using Netflow traces[5]. We have produced 113 traffic matrices, one per day when the network load is the highest. The traffic is aggregated on 15 minutes.

[5]A set of traffic matrices generated from netflow traces has been made publicly available ([11]).

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 loose 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 and 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. At this stage, the aggregated netflow information grouped by ingress node is called the interdomain traffic matrix.

2) Collecting BGP data: BGP is the protocol used for interdomain routing. It relies on TCP sessions to exchange interdomain routes. Sessions between routers of the same Autonomous System (AS) are called iBGP (*internal BGP*) sessions. Sessions between routers belonging to different ASes are called eBGP (*external BGP*) sessions. Routers in a network use these iBGP and eBGP sessions to exchange routes. Each BGP router sends to its peers (iBGP and eBGP sessions) its best route towards all destinations. A router receiving routes through BGP sessions determines its own best routes using the BGP decision process which is made of several criteria (see next subsection for details). Inside an AS, there is usually a full mesh of iBGP sessions, such that each router in the domain knows the routes of all other routers to all destination prefixes. It is the case for the network we consider.

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) 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 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 nodes, from the same peers, with the same BGP parameters[6]), 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.

B. Influence of hot-potato routing

In this section we show that the set of IGP link metrics has an influence on the intradomain traffic matrix and how.

The BGP decision process is made of several criteria:

1) Prefer routes with the highest local preference which reflects the routing policies of the domain;
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;
6) Prefer the route with the lowest IGP distance to the egress point;
7) If supported, apply load sharing between paths. Otherwise, apply a domain-dependant tie-breaking rule, e.g., select the one with the lowest egress ID.

Hot-potato routing occurs when the egress node is selected using the 6th criterion, i.e. to select the route with the lowest IGP distance to the egress point. If for a certain prefix, this criterion was used to select the egress point, a change in the set of IGP link metrics can change the egress node for this prefix. Indeed, if the IGP cost toward an egress node that was not chosen becomes smaller than the cost toward the egress node that was chosen, a traffic shift will occur in the network. In this case, the intradomain traffic matrix changes for involved egress nodes.

Our method of generating the intradomain traffic matrix allows us to plan the traffic shifts that would occur from the change of the IGP link metrics. Indeed, we first generate the (invariant) interdomain traffic matrix which is not influenced by hot-potato routing. When we map the interdomain traffic

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

matrix to the intradomain traffic matrix, we take into account the whole BGP decision process (via C-BGP), including hot-potato routing. If we give C-BGP a new set of link metrics, we will use a new egress point and the resulting intradomain traffic matrix will be updated accordingly. Figure 2 describes this process. We have generated three different intradomain traffic matrices, one considering the actual link metrics, one with inverse capacity link metrics (InvCap) and one with unitary metrics (HopCount). We have noticed that quite huge differences exist between these traffic matrices. Indeed the outgoing traffic of some nodes can differ up to three times if we consider another matrix.

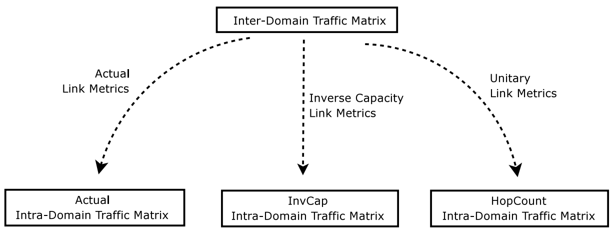
Inter-Domain Traffic Matrix
Actual Link Metrics
Inverse Capacity Link Metrics
Unitary Link Metrics
Actual Intra-Domain Traffic Matrix
InvCap Intra-Domain Traffic Matrix
HopCount Intra-Domain Traffic Matrix

Fig. 2. Intradomain Traffic Matrix generation for various sets of IGP metrics

We have also generated traffic matrices considering the link weights computed by *IGP-WO*. Note that the *IGP-WO* algorithm provides a good set of metrics considering that the intradomain traffic matrix is invariant, which is not the case. It is thus important to compute the resulting Traffic Matrix for each set of weights computed by *IGP-WO*. For each of the 113 Traffic Matrices we have run *IGP-WO* and generated a new Resulting Traffic Matrix. Note that the Hot Potato effects in IGP-WO are considered in more detail in [12].

Let us note that this kind of simulation is only possible with a toolbox like TOTEM which embeds all mentioned traffic engineering algorithms, methods and simulators. It is absolutely not possible to obtain these traffic matrices with other techniques that measure the actual traffic matrix like the ones using Label Switched Path counters or less precise Traffic Matrix inference techniques, or even with the new functionality of latest Netflow version. With such techniques, it is only possible to obtain the actual intradomain traffic matrix (or at least an estimation of it), *but it is not possible to simulate the effect of changing the link metrics on this intradomain traffic matrix.*

To demonstrate the usefulness of using traffic matrices taking into account hot potato effects, let us compute how they differ. We define the distance between two traffic matrices ($D_1(s,t)$ and $D_2(s,t)$) as $\frac{\sum_{(s,t)} |D_1(s,t)-D_2(s,t)|}{\sum_{(s,t)} \frac{D_1(s,t)+D_2(s,t)}{2}}$. Using this distance (which is a kind of percentage of variation), the distance between the Actual traffic matrix and the HopCount or InvCap traffic matrices is between 0.1 % and 65.6%. The mean distance is 9.2% between the Actual and the HopCount traffic matrices and 29% between the Actual and the InvCap TMs. The mean percentage of origin-destination pairs for which corresponding traffic is different is respectively 7.4% and 14.9%.

VI. Traffic and Network State Analysis

A. How to compare different algorithms?

The comparison of different algorithms is not easy and requires some special care. Indeed, all the algorithms do not have the same objectives when optimising routes. Some try to minimise the load of the most loaded link, some try to minimise the length of the paths, while some try to balance the load over the whole network, or to minimise a combination of these objectives, etc. Thus, one algorithm can be the best regarding one criterion, and bad regarding other criteria[7]. To be as objective as possible in our comparison, we look at several criteria. When looking at the results, we think it is important to have in mind a good description of each algorithm, what it is supposed to optimise, and in which case it is supposed to be used.

1) Centralised algorithms: These algorithms have to run on a centralised server which has access to the whole topology and to all traffic data. It is not possible to deploy them in a decentralised on-line way.

a) Multicommodity network flow (MCNF): We have used an arc-node Linear Program (LP) formulation of the routing problem ([14]). One commodity is assigned to each (source, destination) node pair. The value of this commodity is the traffic that flows from the source node to the destination node. Each link of the network is assigned a capacity. The algorithm finds a routing strategy so that the objective function is minimal, while respecting the capacity constraints. As objective function, we have chosen the maximum link utilisation. The algorithm finds the optimal value for this (and only this) objective function. This is useful to obtain an optimal value for one criterion, which can be used to determine the quality of other solutions. We have used GLPK (GNU Linear Programming Kit) to solve the MCNF problem.

b) IGP-WO: The objective function IGP-WO seeks to minimise is equal to the sum over all links of a convex piecewise linear function increasing with the link load, which assigns a very high value to highly loaded links[8]. The idea is that it is cheap to route a flow over a link with a low utilisation. But when the utilisation increases, it becomes more expensive because the buffering delay increases as well but also because the network becomes more sensitive to bursts. Due to the very high penalty for highly loaded links, this objective function will tend to minimise their load. A general advantage to working with a sum rather than a maximum is that even if there is a bottleneck link for which it is impossible to avoid a high load, the objective function still cares about minimising the load of other links of the network.

2) Decentralised algorithms: The algorithms in this section have in common that they are designed to be deployed in a decentralised on-line scheme. To use these algorithms in a centralised scheme, we proceeded as follows. For MPLS algorithms, we compute for each (source, destination) node pair a path and establish this path as an LSP. These paths are computed in sequence, taking already established paths

[7]In [13], we compare different objective functions considering a set of TE metrics.

[8]This objective function is defined in [4].

into account. But we do not change the path of an already computed LSP even if this could lead to a better global optimisation. This implies that the LSPs' establishment order can have an influence on the quality of the solution found. In this section, we suppose that we use MPLS to establish all the computed paths, but any tunnel-based technology is possible.

a) (C)SPFActualmetrics: This algorithm has been explained in section IV-A. It computes the shortest path (under capacity constraints) between two points based on the metrics currently used in the network.

b) (C)SPFHopCount: This algorithm is a (C)SPF algorithm for which the metric is 1 for each link and thus tries to find minimum hop paths. The optimisation idea behind this scheme is that if the paths are short, the mean load and the mean propagation delay should be low as well.

c) (C)SPFInvCap: This algorithm is a (C)SPF algorithm for which the metrics are the inverse of the capacity of the links. This scheme encourages flows to be routed over high capacity links, leading to a low mean link utilisation.

d) CSPFInvFreeBw: This algorithm is a CSPF algorithm for which the metrics are the inverse of the *residual* capacity of the links. Thus, when a new LSP is established, the metric is updated for each link used by this LSP. This scheme is a kind of *CSPFInvCap* algorithm, but which takes into account already established LSPs. We notice that for this algorithm the paths depend on the LSPs' establishment order.

e) DAMOTE: This algorithm has been explained in section IV-C. In this study we have used as cost function a combination of two components. The first component is a shortest path component. The second one, called the *load balancing* component, is the standard deviation of the link utilisations (try to have the same utilisation on all the links of the network). We can set a parameter (α) that specifies the importance of the first component over the second in the score function. $\alpha = 0$ is equivalent to a pure load-balancing score function (no shortest path component). As α increases, the shortest path contribution gets more and more important. We have used $\alpha = 2$ in this study.

B. Results based on a representative snapshot of the current traffic load

In this section, we use the toolbox to compare the different routing algorithms presented in the preceding section. Section VI-B1 presents the notations used in this section. Sections VI-B2 and VI-B3 analyse and compare respectively IP and MPLS solutions. These sections try and determine the best algorithm for the current traffic and the current network. For this purpose these analyses are based on a representative snapshot of the real traffic of the current network. These allow us to compare many different algorithms concerning different criteria such as "Maximal Utilisation", "Mean Utilitation", "Standard Deviation" and "10th Percentile".

1) Notations: To compare all these methods, we analyse the link loads and in particular the maximal link utilisation (i.e. the utilisation of the most utilised link). This value gives some information about the network bottleneck. We also analyse the mean utilisation, the 10^{th} percentile[9] and the standard deviation which reflects the load balance. Results are presented in table I for one typical Traffic Matrix. The maximal utilisation given by the multicommodity network flow (MCNF) algorithm gives the lower bound value we can achieve. This is only valid for the Max Utilisation value because the *MCNF* only optimises this criterion. For *CSPFHopCount*, *CSPFInvFreeBw* and *DAMOTE*, *decr* means a decreasing bandwidth request size establishment order. The *TM* column specifies which Traffic Matrix has been used for the simulation. *A* stands for the actual traffic matrix, *HC* for the HopCount traffic matrix, *IC* for the InvCap traffic matrix (refer to fig. 2) and *R* for the "resulting" traffic matrix (obtained with the metrics optimised by *IGP-WO*). **ECMP* means that in this case the results were identical with or without ECMP.

We have to analyse these results with care. Indeed, the current IGP weights configuration (*SPFActualmetrics* line in the table) combines different objectives like favouring high capacity links while respecting some delay constraints. In these simulations, our only goal is to minimise the link utilisations and to balance the traffic in the whole network.

2) IP Solutions: We have tested IP SPF algorithms with ECMP or not. *SPFActualmetrics* and *SPFInvCap* algorithms lead to the same results with or without ECMP. On this topology and traffic matrix *SPFHopCount* is the only IP algorithm that has different results when ECMP is enabled or not. Let us note that if the routing scheme resulting from a SPF algorithm does not lead to overloaded links, the corresponding CSPF LSP path computation algorithm would lead to the same result (if we do not take into account Hot Potato effects). It is not the case if SPF leads to overloaded links because in this case CSPF will avoid these overloaded links.

One interesting result is that *SPFInvCap* does not provide the optimal value of mean utilisation, while it is supposed to do so (see [13] for details). In fact, this result is true only if the traffic matrix is invariant. In our case, as we take into account the effects of hot-potato routing, this is not true any more as the traffic matrix has changed.

In this network *SPFInvCap* and *IGP-WO* are the best pure IP solutions. These are almost equivalent and improve the network load balance.

3) MPLS Solutions: For all the MPLS algorithms, we use the actual intradomain traffic matrix. Indeed, as MPLS does not change the link metrics, there is no Hot-Potato effect in this case. So, *CSPFInvCap* does provide the optimal value of *Mean Util*, while it was not the case for *SPFInvCap*. But we can still see that the mean utilisation value of *SPFInvCap* is not far from the lowest one. This bad hot-potato effect is quite limited in our simulation but could be more important in other networks. In fact, it depends on the part of the total traffic whose next-hop is chosen via hot-potato routing in the BGP decision process.

Traffic engineering on the particular network we have studied reaches quite rapidly its limits. Due to the high financial cost of the links reaching some nodes, these nodes are connected to the network by two low capacity links (155 Mbps),

[9]The N^{th} percentile gives the utilisation of the link such that N% of the links of the network are more loaded than this link.

Algorithm	Max Util	Mean Util	Std Dev	10^{th} Perc	TM
MCNF	**47.5%**	-	-	-	A
IP Algorithms					
SPFActualmetrics *ECMP	85.9%	**6.9%**	12.1%	**20.0%**	A
SPFHopCount ECMP	84.7%	8.8%	15.4%	26.0%	HC
SPFHopCount ¬ECMP	103.8%	9.2%	18.0%	**20.4%**	HC
SPFInvCap *ECMP	52.1%	7.2%	9.7%	**20.0%**	IC
IGP-WO Resulting TM	52.4%	7.0%	9.8%	**20.4%**	R
MPLS Algorithms					
CSPFActualmetrics	85.9%	**6.9%**	12.1%	**20.0%**	A
CSPFHopCount *decr*	98.5%	9.2%	17.9%	**20.4%**	A
CSPFInvCap	52.4%	**6.8%**	9.8%	**20.0%**	A
CSPFInvFreeBw *decr*	**47.9%**	**6.8%**	9.6%	**20.0%**	A
DAMOTE$_{\alpha=2}$ *decr*	**47.5%**	10.5%	**8.1%**	23.2%	A

TABLE I
COMPARISON OF ALGORITHMS ON THE OPERATIONAL NETWORK WITH A TYPICAL BUSY PERIOD TRAFFIC MATRIX

which really limits the routing possibilities. Indeed it is always these bottleneck links that achieve the maximal utilisation. Moreover, these nodes send a big amount of traffic with respect to the bandwidth of the links they are connected to. Regarding traffic coming in and out of these nodes, there is no other possibility than forwarding it on the low capacity links, leading to a high utilisation of these links. Thus, traffic engineering techniques do not have another choice than balancing the traffic between these links. We can see that *DAMOTE* and *CSPFInvFreeBw* are very good for this point.

As we have already mentioned, the quality of the results of all the MPLS methods can depend on the order of establishment of the LSPs. We have evaluated the following establishment orders:

- Decreasing bandwidth request size order;
- Increasing bandwidth request size order;
- Random bandwidth request size order. We tried more than 100 different random orders and took the mean value for each parameter (max utilisation, mean utilisation, ...).

For *CSPFActualmetrics*, *CSPFHopCount* and *CSPFInvCap*, the establishment order does not change the results (if there is no overloaded links in corresponding SPF algorithm, of course). Indeed, for these schemes, the computed path depends only on link metrics which do not depend on the link loads. Thus computing one path at the beginning of the simulation or at the end will provide the same results, which is not the case for algorithms for which the link metrics depend on the link loads, like *DAMOTE* or *CSPFInvFreeBw*. For these two algorithms, the LSPs' establishment order matters. For the traffic matrix we have used, we have noticed that the decreasing LSPs' establishment order is always the best for the max utilisation criteria, which is quite intuitive. For *CSPFHopCount*, the establishment order can also change the results as corresponding SPF algorithm leads to overloaded links in our case.

DAMOTE and *CSPFInvFreeBW* are very good for the Maximal Utilisation. *CSPFInvFreeBW* gives also excellent results concerning other criteria, including the optimal Mean Utilisation value.

C. Planning for future increasing traffic load

This section provides an extrapolation analysis based on increasing traffic demands. We had to upgrade the capacity of some bottleneck links to be able to route the increased traffic. This is a quite important point as it shows how the TOTEM toolbox can be used to aid network planning/provisioning decisions. This section also identifies some shortcomings of IGP-WO.

Out of the huge amount of netflow data we have obtained, we have selected 113 peak traffic matrices (one per day during more than 3 months). To obtain a higher mean load on the network than with the actual traffic matrix we have scaled all these traffic matrices by a factor 1, 1.2, 1.4, ..., 2.6. As some low capacity links were already almost saturated by the actual traffic matrix, we have upgraded these low capacity links from 155 Mbps to 622 Mbps, as we think every network engineer would do in this case. Indeed, otherwise these links would be immediately saturated. Doing this will also remove the fact that the maximal utilisation was always observed on the same low capacity link.

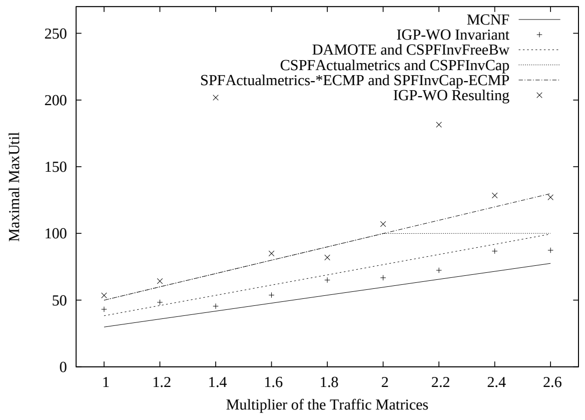

Fig. 3. Maximal link utilisation (worst case scenario over 113 traffic matrices) for selected algorithms on the slightly updated topology (see VI-C)

Figure 3 presents the maximal *MaxUtil* over 113 traffic matrices for selected (best IP and MPLS) algorithms and for all the scale factors. We have reported on this figure *IGP-WO Invariant TM* and *IGP-WO Resulting TM*. *IGP-WO Invariant TM* provides the results as if the TM were invariant (which is erroneous) while *IGP-WO Resulting TM* takes Hot Potato effects into account. Figure 3 shows the following approximate ranking : *MCNF* $<$ *IGP-WO Invariant TM* $<$ *(Damote AND CSPFInvFreeBW)* $<$ *(CSPFInvCap AND CSPFActualmetrics)* $<$ *(SPFInvCap AND SPFActualmetrics)* $<$ *IGP-WO Resulting TM*. The bad position of *IGP-WO Resulting TM* is a very important result. Indeed it highlights the error one can undergo if one does not take Hot Potato effects into account. Note that this is the worst case scenario over all 113 traffic matrices. The high value of the point for scale factor 1.4 is due to a very unfortunate weights setting which causes very unpleasant traffic shifts due to Hot-Potato Routing. This is a quite rare event as we have observed such high value only twice over all our simulations. But this proves that such

Algorithm	Maximal *MaxUtil*		Mean *MaxUtil*	
	TM * 1	TM * 2.6	TM * 1	TM * 2.6
MCNF	29.8%	77.6%	20.6%	53.5%
IP Algorithms				
SPFInvCap ECMP	49.9%	129.8%	32.1%	83.3%
IGP-WO Invariant TM	43.1%	87.4%	31.2%	63.1%
IGP-WO Resulting TM	53.5%	127.1%	31.9%	75.5%
MPLS Algorithms				
Damote	38.3%	99.5%	25.1%	65.2%
CSPFInvFreeBw	38.3%	99.5%	26.3%	65.8%
CSPFInvCap	49.9%	100.0%	32.1%	80.5%

TABLE II
MAXIMAL AND MEAN *MaxUtil* (OVER 113 TRAFFIC MATRICES) ON THE SLIGHTLY UPDATED TOPOLOGY (SEE VI-C)

a big problem can appear in real networks. If we observe the mean value of *MaxUtil* (instead of the maximal value), we observe far better result. Indeed, we observe in this case that the mean *MaxUtil* of *IGP-WO Resulting TM* is always better than *SPFInvCap*. We can also note that in our case, splitting the traffic using ECMP for *SPFInvCap* algorithm slightly decreases the *MaxUtil*, while it has no influence on *SPFActualmetrics* or *SPFHopCount* (this is not shown on the figure).

Concerning pure MPLS solutions, *Damote* and *CSPFInvFreeBW* are clearly the best. We have *MCNF* $<$ *(Damote AND CSPFInvFreeBW)* $<$ *(CSPFInvCap AND CSPFActualmetrics)* $<$ *CSPFHopCount* concerning the maximal *MaxUtil*. *CSPFInvFreeBW* is better than *Damote* concerning the maximal *MeanUtil* while *Damote* is better than *CSPFInvFreeBW* concerning the maximal *10thPercentile*. *Damote* is better than *CSPFInvFreeBW* concerning the mean *MaxUtil* while *CSPFInvFreeBW* is better than *Damote* concerning the mean *MeanUtil*.

Finally, table II presents the values of the maximal and mean *MaxUtil* for the scale factor 1 and 2.6 for all the best IP and MPLS solutions. We can see that if we do not consider Hot Potato effects, *IGP-WO* is a very good solution and is always better than *SPFInvCap*. But if we take those effects into account we see that *IGP-WO Resulting TM* can be in some cases worse than *SPFInvCap*, even if in most of the cases *IGP-WO Resulting TM* is better (the mean *MaxUtil* is lower than for *SPFInvCap*). Table II also shows that all presented MPLS solutions give very good results concerning the mean and maximal *MaxUtil*. Moreover we can see that these algorithms do not lead to overloaded links unlike their corresponding SPF algorithms. All the results and figures of these intensive simulations are available in [15].

D. Conclusion

To conclude this comparison, we can say that MPLS provides very good solutions concerning maximal utilisation while keeping very good performance concerning other criteria. If we had to engineer this network, we would use MPLS with *Damote* or *CSPFInvFreeBw* as routing scheme for the very good solutions these algorithms provide. If the deployment of an MPLS solution is not possible, we recommend *SPFInvCap* with ECMP or *IGP-WO*. If *IGP-WO* is chosen, it is very important to verify that it does not result in the worst case scenario of figure 3 or better, use the technique proposed in [12]. These recommendations are of course only valid for this network. The same kind of study could be performed on another network to decide which algorithm is the best in another situation.

Concerning Hot-Potato routing (HPR), the important point to remember is that disregarding HPR can lead to errors which can be quite high in some cases as it is shown on fig. 3 for IGP-WO. It is important to be aware of this result.

VII. WORST CASE LINK FAILURE ANALYSIS

In this section, we analyse the effects of the failure of each link of the network. For these simulations we consider the real traffic and the base topology (not the upgraded one).

The worst case link failure is defined as follows. S_i is the state of the network when link i is down (S_0 is the state of the network without failure). $\mu(S_i)$ is the maximal link utilisation when the network is in the state S_i. We can say that link i causes the worst case link failure if $\mu(S_i) \geq \mu(S_j), \forall j$.

We proceed as follows. Thanks to TOTEM, we generate an XML scenario file that:

1) Load the topology;
2) Load the actual traffic matrix;
3) Collect information about links' load when there is no failure;
4) Tear down one link;
5) Load the simulated traffic matrix when this link is down. It is important to observe that we take the hot potato effect into account (see section V-B);
6) Collect information about links' load when this link is down;
7) Set up back the link;
8) Repeat the last three steps for every link;
9) Create a chart with the collected information.

This scenario generation can be done in TOTEM easily by typing only one command. The toolbox returns a chart. On this chart, we can see that the maximal utilisation when there is no failure is 85.94% (for link 29). This value reaches 95.06% when link 29 or link 30 fails and a value between 85.94% and 95.06% when link 31 or link 37 fails.

In the topology there is a node (referred to below as node A) that is connected to the rest of the network by two low-capacity links (links 29 and 30).

It turns out that there is a big amount of traffic towards node A (relatively to the capacity of the two links) and that the two links carry only traffic whose destination is node A. So when one of these two links fails, the whole traffic for node A must be carried by the other link and this leads to the worst case link failure.

Another interesting thing we noticed on the chart is that the maximal utilisation decreases relatively to the ”No failure” situation when link 14 or 20 fails. This odd observation can easily be explained as follows. Thanks to TOTEM, we create the following XML scenario:

1) Load the topology;
2) List the shortest paths from all sources to node A;

Failure of link	Link 29		Link 30	
	% of paths	% of traffic	% of paths	% of traffic
None	81.82%	90.4%	18.18%	9.6%
Link 14	68.18%	55.4%	31.82%	44.6%
Link 20	54.55%	50.8%	45.45%	49.2%
Link 31	90.91%	97.05%	9.09%	2.95%
Link 37	86.36%	93.04%	13.64%	6.96%

TABLE III
ANALYSIS OF THE SHORTEST PATHS TO NODE A.

3) Tear down link 14;
4) List the shortest paths from all sources to node A;
5) Set up back link 14;
6) Repeat the last three steps for links 20, 31 and 37.

Writing such a scenario file with TOTEM is quite easy and can be done in a few minutes. The results of the execution of this scenario file are presented at table III. Note that we give the percentage of paths to A using link 29 or 30 (and not the absolute number of paths) for proprietary reasons. Table III also contains information about percentage of traffic using links 29 and 30 and whose destination is node A.

Table III shows that the traffic distribution between links 29 and 30 is bad (more than 90% of traffic uses link 29). Though the presented results are particular to the traffic matrix we used, we can say that we will never reach a perfect load balance of traffic (50% of traffic on each link) whatever the traffic matrix. Indeed, the paths distribution is also very bad (more than 80% of paths use link 29). And, with the routing scheme currently used in the operational network (i.e. IP static SPF), the paths do not change if the traffic changes.

The low maximal utilisation when link 14 or 20 fails can thus be explained by the fact that less paths and less traffic to node A use link 29 and so the utilisation of this link (the most utilised link in the ”No failure” situation) decreases. Similarly, when link 31 or 37 fails, the number of paths and the percentage of traffic to node A using link 29 increase and the utilisation of this link also increases.

VIII. CONCLUSION

This paper demonstrated the usefulness of the TOTEM toolbox. It gives an overview of the answers the toolbox can provide to some important questions a network operator may have. Using the toolbox, an operator can for example see whether his network is well-engineered or not, evaluate the impact of hot-potato routing on the traffic matrix, compare a wide range of IP and MPLS solutions and choose the best in this large set, or see whether the network is enough provisioned to support failures or not.

Another important result of this paper is that we have also shown how the TOTEM toolbox can simulate the traffic shifts that occur in the traffic matrix when the IGP link metrics are updated. These results highlight that updating the link metrics can have undesirable effects on traffic. This is a good example of the interaction between IGP and BGP in real networks. Such interactions can only be simulated with a tool like the TOTEM toolbox which federates both intradomain and interdomain traffic engineering techniques and algorithms.

ACKNOWLEDGMENTS

This work was partially supported by the Walloon Region (Belgium) in the framework of the WIST programme (TOTEM project) and by the European Union under the E-Next and ANA projects. We are grateful to Steve Uhlig (UCL, Belgium) for aggregating the Netflow traces and to Bruno Quoitin (UCL, Belgium) for the clustering scripts and his help on using C-BGP.

REFERENCES

[1] “http://totem.run.montefiore.ulg.ac.be.”
[2] G. Leduc, H. Abrahamsson, S. Balon, S. Bessler, M. D’Arienzo, O. Delcourt, J. Domingo-Pascual, S. Cerav-Erbas, I. Gojmerac, X. Masip, A. Pescaph, B. Quoitin, S. Romano, E. Salvatori, F. Skivée, H. Tran, S. Uhlig, and H. mit, “An open source traffic engineering toolbox,” *Computer Communications*, vol. 29, no. 5, pp. 593–610, March 2006.
[3] S. Balon, L. Mélon, and G. Leduc, “A scalable and decentralized fast-rerouting scheme with efficient bandwidth sharing,” *Elsevier Computer Networks*, vol. 50, no. 16, pp. 3043–3063, Nov. 2006.
[4] B. Fortz and M. Thorup, “Internet Traffic Enginnering by Optimizing OSPF Weights,” in *Proc. IEEE INFOCOM 2000*, 2000, pp. 519–528.
[5] J. L. Marzo, E. Calle, C. Scoglio, and T. Anjali, “QoS Online Routing and MPLS Multilevel Protection: A Survey,” *IEEE Communications Magazine*, october 2003.
[6] W. Grover, *Mesh-Based Survivable Networks, Options and Strategies for Optical, MPLS, SONET, and ATM Networking.* Prentice Hall PTR, 2003.
[7] R. Bhandari, *Survivable Networks : Algorithms for Diverse Routing*. Kluwer Academic Publishers, 1999.
[8] F. Blanchy, L. Mélon, and G. Leduc, “A Preemption-Aware On-line Routing Algorithm for MPLS Networks,” *Telecommunication Systems*, vol. 24, pp. 187–206, 2-4, Oct.-Dec. 2003.
[9] 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.
[10] B. Quoitin, C. Pelsser, O. Bonaventure, and S. Uhlig, “A performance evaluation of BGP-based traffic engineering,” *International Journal of Network Management (Wiley)*, vol. 15, no. 3, May-June 2005.
[11] S. Uhlig, B. Quoitin, S. Balon, and J. Lepropre, “Providing public intradomain traffic matrices to the research community,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 83–86, January 2006.
[12] S. Cerav-Erbas, O. Delcourt, B. Fortz, and B. Quoitin, “The Interaction of IGP Weight Optimization with BGP,” in *Proceedings of ICISP (International Conference on Internet Surveillance and Protection)*, Cap Esterel, France, August 2006.
[13] S. Balon, F. Skivée, and G. Leduc, “How Well Do Traffic Engineering Objective Functions Meet TE Requirements?” in *Proceedings of IFIP Networking 2006, Coimbra*, vol. 3976. Springer LNCS, May 2006, pp. 75–86.
[14] R. Ahuja, T. Magnanti, and J. Orlin, *Network flows, theory, algorithms and applications*. Prentice Hall, 1993.
[15] S. Balon, J. Lepropre, O. Delcourt, F. Skivée, and G. Leduc, “Applying the TOTEM toolbox on an Operational Network,” ULg, Université de Liège - RUN, Research Unit in Networking, Tech. Rep., 2006, available at http://www.run.montefiore.ulg.ac.be/~balon/RUN-TR06-01.pdf.

Simon Balon , graduated as a computer science engineer at the University of Liège (Belgium) in 2003. In August 2003, he joined the Research Unit in Networking of the University of Liège headed by Guy Leduc. Since October 2004, he is PhD student funded by the FNRS. His fields of research are traffic engineering and fast recovery in the context of MPLS.

Jean Lepropre received a 4-year bachelor degree in computer science from the University of Liège (Belgium) in 2004. In September 2004, he joined the Research Unit in Networking of the University of Liège headed by Guy Leduc. In 2006, he got an inter-university Master in Computer Science from the University of Lige. His fields of research are traffic engineering and application of automatic learning methods to computer networks.

Olivier Delcourt graduated as a computer science engineer at the University of Liège (Belgium) in 2003. In August 2003, he joined the Research Unit in Networking of the University of Liège headed by Guy Leduc. His fields of research are traffic engineering and in particular intradomain interdomain interplay.

Fabian Skivée received a Master in computer science from the University of Liège, Belgium, in 2002. In September 2002, he joined the Research Unit in Networking of the University of Liège headed by Guy Leduc. His current research interests are in traffic engineering strategies for IP/MPLS network, heuristic optimisation methods and network fairness.

Guy Leduc , IEEE member, is full professor in the Department of Electrical Engineering and Computer Science of the University of Liège (ULg), Belgium, and is head of the Research Unit in Networking (RUN). He is also part-time professor at the University of Brussels (ULB). His main current research areas are traffic engineering, mobile communication, congestion control and autonomic networking. He is one of the main designers of the ISO E-LOTOS language, and in particular of its formal timed semantics. Since 2007 he is chairing the IFIP Technical Committee (TC6) on Communications Systems, and has been the chairman of IFIP WG6.1 from 1998 to 2004. He is also an executive committee member of the IEEE Benelux Chapter on Communications and Vehicular Technology, and a member of the editorial board of the Computer Communications journal.