Abstract:
This document, which is an update of the previous D41 deliverable, describes problems, approaches and solutions in the area of network management, from the TIGER2 point of view. Besides solution design, it provides detailed evaluations of the proposals showing the feasibility of self-* paradigms when addressing network control and management issues. Higher manageability issues are tackled in six areas. 1) The hitless maintenance approach eases the work of the human operator by mastering maintenance tasks through automated processes. 2) The approach for energy aware routing exploits ideas of flow aggregation, and aims at finding the optimal type and amount of resources for balanced network performance and power efficiency. 3) OPEX and CAPEX awareness through a new networking architecture is described through the LOCARN concept of TIGER2. 4) Further self-optimized network resource allocation can be achieved by fine-tuning the characteristics of the Spanning Tree Protocol. 5) Inter-domain traffic engineering problems are proposed to be tackled through sharing the intelligence between control planes. 6) Finally, more knowledge could also be gathered through extensive traffic monitoring, which allows traffic mix and traffic matrix analysis (among others) to support decision making in order to actuate proper modifications at the network node level for optimal operation.

Editor:
Samir Ghamri-Doudane – Alcatel-Lucent Bell Labs France
Authors:
Samir Ghamri-Doudane, Laurent Ciavaglia – Alcatel-Lucent Bell Labs France
Dario Rossi – Telecom ParisTech
Lluís Fàbrega i Soler - Universitat de Girona
Rémi Clavier – France Telecom Orange Labs
Csaba Simon – Budapest University of Technology and Economics
Pal Varga – AITIA International

Reviewer:
Laurent Ciavaglia – Alcatel-Lucent Bell Labs France

Date of publication: December 2010
# Table of Contents

1 EXECUTIVE SUMMARY ........................................................................................................... 6

2 HITLESS MAINTENANCE ...................................................................................................... 8

2.1 Problem Statement ............................................................................................................ 8
2.2 OSPF Graceful Restart ........................................................................................................... 9
2.3 Maintenance Process .......................................................................................................... 10
2.4 Planning Algorithm and Metrics ....................................................................................... 11
2.5 Metric Choice .................................................................................................................... 12
2.6 Evaluation Results through Simulations ............................................................................. 13
2.7 Evaluation Results using the Emulation Platform ............................................................... 14
2.7.1 COST OF THE PLANNING ALGORITHM ........................................................................ 14
2.7.2 PERFORMANCE OF THE PLANNING PROCESS ....................................................... 15
2.8 Conclusion ........................................................................................................................ 16

3 ENERGY-AWARE ROUTING: A REALITY CHECK .............................................................. 17

3.1 Context ................................................................................................................................ 17
3.2 Energy Model ....................................................................................................................... 18
3.3 Problem Formulation .......................................................................................................... 19
3.4 Experimental Results ......................................................................................................... 21
3.4.1 WORST-CASE SCENARIO ............................................................................................ 21
3.4.2 QoS CONSIDERATIONS ............................................................................................... 24
3.4.3 SENSITIVITY ANALYSIS .............................................................................................. 26
3.5 Conclusion ........................................................................................................................ 27

4 LOCARN .............................................................................................................................. 29

4.1 Problem Statement ............................................................................................................ 29
4.2 Solution .............................................................................................................................. 30
4.2.1 FRAME ............................................................................................................................ 30
4.2.2 NODE ............................................................................................................................. 31
4.2.3 LOCARN FRAME PROCESSING .................................................................................... 34
4.2.4 OTHER ISSUES ............................................................................................................ 34
4.3 Preliminary evaluation results ............................................................................................ 35
4.3.1 LOCARN POC IMPLEMENTATION .............................................................................. 35
4.3.2 LOCARN COMPLIANCE WITH AUTONOMIC PROPERTIES .................................. 36
5 ADAPTIVE PROBABILISTIC FLOODING FOR MULTIPATH ROUTING IN LOCARN ................................................................. 37

5.1 Related work .................................................................................................................................................. 37
5.2 Path discovery algorithm ............................................................................................................................. 38
5.2.1 OVERVIEW ........................................................................................................................................... 38
5.2.2 PRIMARY AND SECONDARY PATHS .................................................................................................... 39
5.2.3 METRICS ................................................................................................................................................ 40
5.2.4 PSEUDOCODE .................................................................................................................................... 40
5.3 Performance evaluation ............................................................................................................................... 42
5.3.1 OVERHEAD ........................................................................................................................................ 44
5.3.2 PATH QUALITY .................................................................................................................................. 47
5.4 Conclusions ............................................................................................................................................... 47

6 SELF-OPTIMIZATION OF NETWORK RESOURCE ALLOCATION IN MSTP ...... 49

6.1 Optimization of network resource allocation ............................................................................................. 49
6.2 The proposed scheme for self-optimization ................................................................................................. 49
6.3 Optimization model for MSTP based technologies .................................................................................. 50
6.4 Evaluation results ...................................................................................................................................... 53

7 INTER-DOMAIN TRAFFIC ENGINEERING FOR BALANCED NETWORK LOAD 55

7.1 Network model ............................................................................................................................................ 55
7.1.1 REFERENCE NETWORK MODEL ...................................................................................................... 55
7.1.2 CORE NETWORK MODELS ............................................................................................................... 56
7.1.3 INVESTIGATED NETWORK TOPOLOGIES ..................................................................................... 56
7.2 The proposed solution: Inter-domain TE cooperation .............................................................................. 57
7.3 Proposed simulation model .......................................................................................................................... 58
7.3.1 TRAFFIC MODEL .............................................................................................................................. 58
7.3.2 SPANNING TREES IN THE AGGREGATION DOMAIN ..................................................................... 59
7.4 Planned work ............................................................................................................................................. 59
7.4.1 DUAL RING CORE TOPOLOGY ....................................................................................................... 59
7.4.2 LOAD BALANCING AS A RESULT OF INTER-DOMAIN COMMUNICATION .................................. 60
7.5 Conclusion ................................................................................................................................................. 61

8 TRAFFIC ANALYSIS FOR THE KNOWLEDGE PLANE .............................................................................. 62

8.1 Problem Statement .................................................................................................................................... 62
8.2 Solution ...................................................................................................................................................... 63
8.2.1 THE MONITOR PLANE ...................................................................................................................... 63
8.2.2 BASIC FUNCTIONS OF THE PROBES ............................................................................................. 64
8.2.3 TRAFFIC PROCESSING ..................................................................................................................... 65
8.2.4 DECISION MAKING ........................................................................................................................ 66
1 Executive Summary

One of the main objectives of the TIGER2 WP4 work is to identify and solve concrete operational scenarios related to network control and management. The first WP4 deliverable (D40) has discussed the rationale behind the TIGER2 approach, addressed its implementation, and detailed the list of study cases that have been identified for further investigation [8]. Then, the next deliverable (D41) has presented the solutions that have been proposed to solve these relevant study cases, and has also outlined their use of self-* paradigms [38]. Finally, this deliverable, which is an updated version of D41, augments these solutions with a detailed evaluation study. The foreseen results aim at assessing the feasibility and performance of the proposed approaches, and hence self-* paradigms, in tackling network control and management issues.

There are altogether six areas of higher network and service manageability addressed in this document. The following presents briefly the technical problems tackled by each of these study cases.

The Hitless Maintenance approach proposes a solution for the automation and optimized orchestration of maintenance operations for IP/MPLS networking elements, based on an adaptive and fully-distributed planning process. It allows the operator to focus on productive activities, since he/she is greatly relieved from tedious maintenance tasks.

Energy aware routing is the second topic of this document. One of the most common practices for acting in a green fashion in network dimensioning consists in resource consolidation. This technique aims at reducing the energy consumption due to devices underutilized at the considered interval of time. The solution aims for an optimal balance, where the required level of performance will still be guaranteed, but using an amount of resources that is dimensioned for the current network traffic demand rather than for the peak demand (or more). Flow aggregation may be achieved, for example, through a proper configuration of the routing weights.

Higher manageability should be considered from the OPEX and CAPEX point of view, since theoretically feasible solutions often mean practically unrealistic possible investments. LOCARN (Low Opex and Capex Architecture for Resilient Networks) is an imaginative, new network paradigm, which aims to explore two concepts (auto-forwarding and enhanced broadcast) in order to increase as much as possible network simplicity and hence increase savings in both OPEX and CAPEX domains. Chapter 4 of this document aims to offer a sufficient enough in-depth view of the LOCARN network concepts with the objective to prepare its Proof-Of-Concept implementation. Then, Chapter 5 focuses on describing and assessing a distributed source routing algorithm for topology discovery with the aim to improve the performance of the LOCARN architecture. It proposes a plug-and-play control plane, able to find multiple paths toward the same destination, and introduce a novel algorithm, called adaptive probabilistic flooding, to achieve this goal.

Self-optimized network resource allocation in MSTP (Multiple Spanning Tree Protocol) takes a deep dive into management algorithms. Self-optimization of network resource allocation aims
to provide automatically without human intervention an optimal allocation. Periodically an optimization model is run, then the calculated optimal resource allocation is compared with the actual one, and if their difference is large enough, reconfiguration of network resources is initiated. The goal of the optimization model can be either minimizing the number of allocated spanning trees or the amount of allocated capacity.

Another aspect of network manageability appears at inter-domain areas. Chapter 6 of this document proposes new traffic engineering methods for balanced network load. The main idea of the solution is to use shared intelligence between control planes, where the core intra-network functions are unchanged and only the inter-network control planes co-operate which enhances the performance.

The knowledge plane concept allows self-management networks and networking services by applying sensors, processing their data, making (probably fine-tuning) decisions based on the processing results, and applying the corrective steps at the network or servicing node level. The elementary steps toward the solution consist of gaining knowledge about network status and traffic characteristics is to gather and process such data, which then provide a basis to trigger corrective actions. The Monitor plane concept, and proposed physical equipment (including SGA10GED, a network interface card developed inside TIGER2), is introduced in in the final technical chapter.

**Note:** The D40 deliverable contains the description of an additional study case entitled: "Adaptive control of Path Computation Elements" [8]. This study case shall be considered only as an example of a concrete operational scenario related to network control and management. It has not been investigated further due to budget restrictions in Spain. On the other hand, the D40 deliverable did not initially provide the description of the "Traffic analysis for the knowledge plane" study. It has been identified a posteriori since it is a common requirement shown by several of the initial WP4 study cases.
2 Hitless Maintenance

The maintenance of communication systems is a critical operation which monopolizes human and network resources. The management of multiple, parallel maintenance jobs is a complex task that can generate faults and undesirable service interruption. In the context of IP/MPLS networks, Graceful Restart mechanisms allow, under strict conditions, the maintenance of a single router without impacting its forwarding plane. Still, the network-wide coordination of the routers restarts is an unresolved problem. To solve this issue, the “Hitless maintenance” study case has been defined [8]. It proposes a solution for the automation and optimized orchestration of maintenance operations, based on an adaptive and fully-distributed planning process. The operator is then relieved from tedious maintenance tasks and is only responsible for setting performance objectives and assessing progress reports.

2.1 Problem Statement

Maintenance operations are frequent tasks in telecommunication networks, and cover a large panel of hardware and software interventions. Although necessary, these operations result, most of the time, in partial or complete service interruption, which is detrimental to the end user and to the network operator.

Additionally, network-wide maintenance activities can rapidly become a complex task preempting precious human skills and time. They are error-prone as well, and one of the main causes of network downtime [1]. As a result, the maintenance process is a major generator of costs, and constitutes a relevant example of the increasing complexity in operating today’s networks.

Consequently, a key challenge is to relieve human operators from tedious maintenance tasks, to minimize service disruptions, and hence to drive down the costs. This is part of the Opex challenge that is pushing towards the introduction of trusted autonomic processes within network operations.

Technically speaking, the challenge is to provide network operators with a solution for the automation and optimized orchestration of maintenance operations, with minimal impact on the network. The operator is thus only responsible for the publication of maintenance targets, the setting of high level objectives and the track of progress reports.

Within this context, we focus here on the maintenance of the control plane functionalities in IP/MPLS transport networks by capitalizing on the standardized graceful restart mechanisms [2][3]. These mechanisms are enabled by the physical separation of the control and data plane features in today’s routers. Indeed, it is possible to restart the control software while the data plane functionalities are still up and running. Therefore, the router can be kept within the forwarding path during this maintenance period. The OSPF graceful restart mechanism [2] has thus no impact on the traffic and aims at minimizing the need for routing re-convergence and, by the same, at optimizing the usage of the network resources. It can be used to implement several maintenance activities such as: software upgrades or patching, error corrections, re-initializations and even hardware interventions.

However, in case of network-wide maintenance activities, human operators are still required in order to manually start and drive the maintenance process of each targeted component, and this is generally done in a sequential way. It is hence very consuming in terms
of time, manpower and costs. This study case proposes a solution that completely automates the planning and implementation of such maintenance activities through distributed processes. Nevertheless, the operator remains in the network control loop as he defines the performance objectives and can continuously check their achievement.

### 2.2 OSPF Graceful Restart

Graceful Restart (GR) is a set of mechanisms developed at the IETF for signaling and routing protocols such as LDP [3], and OPSF [2]. The OSPF-GR technique enables to restart the routing process on an individual node or interface without interrupting the overall network service. This mechanism is also called “non-stop forwarding” since the data plane of the restarting node can run, for a short period of time, out of synchronization from the control plane and therefore can continue to forward the traffic normally through the node. This behavior is feasible thanks to the physical separation of the control and forwarding functions as it is the case in most of today’s carrier-class routers.

Usually, when a given router interrupts its routing process, the neighboring nodes trigger automatically a re-convergence of the topology to reflect the router unavailability. OSPF-GR adapts this default behavior of the OSPF protocol and allows the restarting router to remain visible thanks to the coordinated action of its neighbors. However, in order to work properly, the network topology must remain stable and the restarting router must maintain the integrity of its forwarding table across the graceful restart procedure. In case network topology changes are detected, the normal OSPF restart will take over for safety reasons (routing loop avoidance).

The OSPF router performing the maintenance operation is designated as the Restarting router whereas its directly connected neighbors are called Helper routers. These nodes must cooperate in order to achieve a graceful restart. RFC3623 [2] details the conditions and responsibilities for a router running in helper mode, in addition to the operation of the restarting router. The overall process is summarized in Figure 2.1.

The main constraint and particularities that characterize the functioning of the OSPF Graceful Restart mechanism are:

- A single router can simultaneously serve as a helper for multiple restarting neighbors.
- A router cannot perform a graceful restart while it acts as helper for its neighbor(s).
- If the restarting router is normally reloaded and flushes its Grace-LSAs within the grace period, the graceful restart is considered as successfully terminated.
- However, if the grace period expiries, the helpers revert back to a standard OSPF behavior.
2.3 Maintenance Process

The graceful restart mechanism allows performing maintenance operations on the control plane of network equipments with a minimal impact on the traffic. These maintenance operations ranges from simple restarts and software upgrades to hardware interventions. Even if it is not advised, graceful restarts can be used for unplanned outages (such as the crash of a router's control software, an unexpected switchover to a redundant control processor, etc).

Based on these mechanisms, we propose here a solution that completely automates the maintenance process in network-wide environments. Figure 2.2 illustrates the proposed concept. The role of the operator is to simply publish maintenance jobs (targeted equipments and required procedures), define high-level performance objectives and specify safety parameters. The computation and implementation of the maintenance plan is done within the network elements through distributed and cooperative processes. Finally, the operator has the ability to track live progress and completion reports, which keeps him within the control loop of its network.
Figure 2.2 - Automating the maintenance process

The implementation of this solution requires addressing the following key features:

- Providing relevant management tools: This includes the implementation of management interfaces and communication platforms that allow operators to publish maintenance targets, to enforce high level objectives and then to track progress reports. It should be also used by operators to specify safety parameters that allow stopping the automated processes and reverting back to manual procedures, in order to avoid cascading failures and guarantee the network consistency. A relevant candidate option to implement the above communication requirements and interfaces is the use of publish/subscribe systems. These can be instantiated through either centralized or distributed repositories.

- Empowering the network: This is a key enabler for autonomic behaviors. Besides the extraction of dependency models and technical constraints that shall drive the planning of maintenance actions, it is mainly about the design and implementation of distributed planning algorithms that are driven by the functional constraints as well as the performance objectives enforced by the operator.

The reminder of this chapter (solution description) focuses on the proposal and assessment of such a distributed process in the case of domain-wide control plane maintenance using OSPF graceful restarts. The objective is to compute and implement, in a distributed way, the optimal maintenance plan that complies with the operator’s objectives and underlying technical constraints. It is actually an election process that designates the candidate routers at each maintenance step.

2.4 Planning Algorithm and Metrics

The characteristics of the distributed planning algorithm that should be implemented by all the network nodes during the election process are as follows:

- Compliance with the protocol constraints in terms of graceful restart (previously outlined in section 2),

- Support of multiple election metrics in order to fulfill the operator directives and performance objectives,
• Minimizing the communication overhead necessary to complete the election process,
• Ensuring the convergence of the election process in a finite and short period of time.

Figure 2.3 - Pseudode of the distributed planning algorithm (one round)

A version of such a distributed algorithm is detailed in Figure 2.3. At each maintenance step, this algorithm allows to elect one candidate node per neighborhood. The choice of the appropriate nodes is based on a decision metric. In case of equality between the metric of two or several nodes, an arbitrary deterministic parameter is used to break indecisions (for example, a hash value of the router identifiers). Of course, multiple metrics are possible for such an election; the relevant metric should be chosen according to the performance objective set by the operator.

Furthermore, when implementing the proposed algorithm, each node communicates only with its direct neighbors and, based on the exchanged metrics, the appropriate nodes are designated for maintenance. Therefore, the communication overhead is very limited and the convergence time is bounded by the depth of the network.

2.5 Metric Choice

It is important to remind that the main constraint when implementing graceful restarts is the impossibility for two adjacent nodes to simultaneously perform a graceful restart.
Consequently, the planning of maintenance operations can be modeled as graph coloring problem [4]. Based on this, we propose a first set of possible metrics that shall be considered in the assessment study. In the following, these are described along with their expected behavior and performance results:

- **Maximum Degree**: In this case, the election process designates first the nodes with the highest degree (number of one-hop neighbors). This is a well known simple heuristic [5] to minimize the number of rounds necessary to perform the maintenance and restart of all the routers, which will thus minimize the overall maintenance time.

- **Minimum Degree**: Oppositely, the election process designates first the nodes with the lowest degree. Using this strategy, the maintenance process focuses first on the routers within the periphery of the network and then proceeds to the more central ones.

- **Random Metric**: The election process is here based on an arbitrary criterion which allows to fairly distribute the position of restarting routers within the network topology. The results of this metric can be used as a comparison reference as well.

- **Maximum 2-hop Degree**: The metric is computed as the degree of the node plus the mean degree of its one-hop neighbors. It is a more complex metric, in the sense that its computation is less straightforward than the previous ones. It also tends to minimize the overall maintenance time.

### 2.6 Evaluation Results through Simulations

In order to evaluate the previous metrics and confirm their expected behavior, we have run a first set of experiments using realistic ISP topologies [6]. The proposed election process and candidate metrics are implemented among the nodes of these topologies with a target to perform a network-wide maintenance. The values of the arbitrary deterministic parameter, that are used to compute the random metric and to break indecisions, are changed in each experiment. This leads to a variation of the obtained performance results.

Table 2.1 shows the mean number of maintenance rounds that are necessary to address all the network routers for six different topologies. The metrics based on maximum degrees confirm their performance in term of total maintenance time, even if the optimality is not guaranteed. Indeed, minimizing the number of rounds (graph coloring) is NP-complete [4].

<table>
<thead>
<tr>
<th>Topology Size</th>
<th>Random</th>
<th>Min Degree</th>
<th>Max Degree</th>
<th>Max 2-hop Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>79 NODES</td>
<td>4.56</td>
<td>4.58</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>87 NODES</td>
<td>4.67</td>
<td>5.66</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>108 NODES</td>
<td>5.3</td>
<td>5.86</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>141 NODES</td>
<td>5.71</td>
<td>6.98</td>
<td>4.82</td>
<td>5</td>
</tr>
<tr>
<td>161 NODES</td>
<td>5.49</td>
<td>5.64</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>315 NODES</td>
<td>7.11</td>
<td>8.08</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>
Besides this, the completion ratios after each maintenance round are plotted in Figure 2.4. These are mean values for all the tested topologies and relative experiments. The "minimum-degree" metric shows very high completion ratios during the first maintenance rounds. However, the central nodes, which are strongly connected, are addressed almost sequentially at the end of the maintenance process. This trend significantly increases the total maintenance time. The "maximum 2-hop degree" and "random" metrics show trade-off performance results and hence represent relevant alternatives when multiple objectives are expressed.

2.7 Evaluation Results using the Emulation Platform

In order to deepen the evaluation of the proposed maintenance platform and demonstrate its feasibility, an emulation platform has been developed. It is based on the JADE multi-agent framework, where each network node is implemented using two complementary agents: the graceful restart agent and the planning agent. Indeed, a realistic OSPF graceful restart behavior, the previously detailed planning algorithm and three relevant metrics (maximum degree, minimum degree and random metrics) have been implemented. The emulation platform is described in details in the D51 deliverable [7].

The following discusses first the cost of the planning algorithm, under real time conditions, in terms of convergence time and number of exchanged messages (bandwidth cost). Then, the performance of the overall planning process is summarized with the aim to confirm the results and conclusions that has been provided in the previous section.

2.7.1 Cost of the planning algorithm

As stated in section 2.4, the communication overhead, due to the planning algorithm, is very limited and its convergence time is bounded by the depth of the network. In order to quantify and confirm this assertion, the following cost parameters have been studied:
• **Time cost**: It is computed as the mean convergence time of the election phase per node for each maintenance round.

• **Bandwidth cost**: It is computed as the mean number of exchanged election messages per node during the overall maintenance process. It comprises the messages for metric exchange and decision notifications necessary to achieve the election process.

These two parameters have been computed while emulating four realistic ISP topologies of 74, 79, 141 and 161 nodes respectively. The obtained results are reported in table 2.2 and table 2.3 for each of the implemented metrics. These results confirm the rapid convergence of the planning process as well as its very low cost in all the emulated environments. The "minimum degree" metric show the lowest costs as a large proportion of the nodes perform their maintenance during the first rounds. Furthermore, the messages exchanged during the election phase can be embedded within regular OSPF messages. The cost of the planning algorithm is indeed negligible.

**Table 2.2** – Mean convergence time (in milliseconds) of the election phase per node

<table>
<thead>
<tr>
<th>Topology Size</th>
<th>Random</th>
<th>Min Degree</th>
<th>Max Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>74 NODES</td>
<td>192.78</td>
<td>122.17</td>
<td>275.52</td>
</tr>
<tr>
<td>79 NODES</td>
<td>202.69</td>
<td>137.69</td>
<td>219.26</td>
</tr>
<tr>
<td>141 NODES</td>
<td>187.47</td>
<td>139.86</td>
<td>265.34</td>
</tr>
<tr>
<td>161 NODES</td>
<td>294.61</td>
<td>148.72</td>
<td>361.22</td>
</tr>
</tbody>
</table>

**Table 2.3** – Mean number of messages per node

<table>
<thead>
<tr>
<th>Topology Size</th>
<th>Random</th>
<th>Min Degree</th>
<th>Max Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>74 NODES</td>
<td>3.94</td>
<td>3.81</td>
<td>4.09</td>
</tr>
<tr>
<td>79 NODES</td>
<td>4.07</td>
<td>3.82</td>
<td>4.17</td>
</tr>
<tr>
<td>141 NODES</td>
<td>4.05</td>
<td>3.93</td>
<td>4.18</td>
</tr>
<tr>
<td>161 NODES</td>
<td>3.86</td>
<td>3.73</td>
<td>3.9</td>
</tr>
</tbody>
</table>

**2.7.2 Performance of the planning process**

Figure 2.5 depicts the evolution, in time, of the maintenance process (completion ratio) for each of the four emulated topologies and implemented metrics. Moreover, the provided curves highlight the time at which the overall maintenance process reaches 50%, 75% and 100% completion ratios. More details regarding the behavior of the emulation platform and ensuing results can be found in the D52 deliverable [72].

These results confirm the previous conclusions: the "maximum degree" metric minimizes the overall maintenance time, the "minimum degree" metric shows very high completion ratios during the first rounds, while the "random" metric represent a tradeoff.
2.8 Conclusion

Although necessary, maintenance operations are generally error-prone, time and effort consuming tasks for network operators. They thus contribute to the increase of operational expenditure. To address this issue, the proposed study case provides a framework that automates the maintenance process while it keeps the operator within the network control loop. This framework is applied to the maintenance of network control plane with a use of OSPF graceful restarts in order to minimize the impact on the traffic. A distributed planning process and candidate metrics have been designed in order to automate the implementation of operator's objectives. Then, an evaluation of the planning process performance and cost has been carried out using simulations, as well as the developed emulation platform. The obtained results demonstrate the feasibility of the proposed platform (planning algorithm and metrics).

Besides, one of the main challenges of the proposed maintenance solution is to translate the operator's objectives into appropriate election metrics and processes within the network. The studied metrics are relevant candidates to address the operator requirements, and their priorities, in terms of total maintenance time and execution order, as confirmed by the obtained results.
3 Energy-Aware Routing: a Reality Check

In this chapter, we analyze the design of green routing algorithms and evaluate the achievable energy savings that such mechanisms could allow in several realistic network scenarios. We formulate the problem as a minimum energy routing optimization, which we numerically solve considering a core-network scenario, which can be seen as a worst-case for energy saving performance (as nodes cannot be switched off). To gather full-relief results, we analyze the energy savings in various conditions (i.e., network topology and traffic matrix) and under different technology assumptions (i.e., the energy profile of the network devices).

These results give us insight into the potential benefits of different “green” technologies and their interactions. In particular, we show that depending on the topology and traffic matrices, the optimal energy savings can be modest, partly limiting the interest for green routing approaches for some scenarios. At the same time, we also show that the common belief that there is a trade off between green network optimization and performance does not necessarily hold: in the considered environment, green routing has no effect on the main network performances such as maximum link utilization.

3.1 Context

Consciousness on energy consumption is nowadays rising in the ICT field, the network being an important contributor to the total power consumption, and probably the one with the highest foreseen rising rate: the effort of bringing energy-awareness in network elements and processes is usually referred to as green networking.

Once a network has been designed (i.e., the resources that will compose it have been deployed), a periodical off-line process is applied to optimize the utilization of resources, which we will refer to as “routing optimization”. This classical process consists in particular in determining the paths used for each origin-destination pair or, equivalently, to ingress-egress routers in a transit network. Common optimization objective is to avoid congestion by e.g., balancing the traffic as evenly as possible on the network links, or by ensuring that maximum link utilization always remains below a given threshold. In pure IP networks, the path used by each flow is determined by the Internal Gateway Protocol (IGP), based on link administrative weights. Network dimensioning is thus handled by careful weight assignments, for instance using IGP Weight Optimization (IGP-WO) algorithms [39].

One of the most common green practices in network dimensioning consists in resource consolidation: this technique aims at reducing the energy consumption due to devices underutilized at a given time. Given that the traffic level in a given network approximately follows a well known daily and weekly behavior, there is an opportunity to aggregate traffic flows over a subset of the network devices and links, allowing other devices to be temporarily switched off. This solution shall of course preserve connectivity and Quality of Service (QoS), for instance by limiting the maximum utilization over any link. In other words, the required level of performance will still be guaranteed, but using an amount of resources that is dimensioned over the actual traffic demand, rather than for the peak demand. Flow aggregation may be achieved, for example, through a proper configuration of the routing weights in an IP network.

This approach has been evoked in [40] as a hypothetical working direction. The authors of [41] take a first practical step in this direction, with the proposal and evaluation of some
greedy heuristics, which are based on the ranking of nodes and links with respect to the
amount of traffic that they would carry in an energy-agnostic configuration. In this work, we
instead formulate the green routing as an optimization problem, which we numerically solve to
evaluate the achievable energy savings. Aiming at a realistic evaluation, we consider (i) several
power models corresponding to different technologies, (ii) an actual network topology and (iii)
real traffic matrices, taken from an operational network. At the same time, we take a
deliberately conservative approach by choosing a core-network where, since all nodes generate
and receive traffic, no node can be turned off – which constitutes a realistic worst-case
scenario for network resource consolidation scheme. Our goal is to get insight into potential
energy savings in realistic scenarios, and to identify room for future improvement.

3.2 Energy Model

In order to evaluate the energy saving of our green solution, it is fundamental to rely on an
accurate energy consumption model. Yet, we point out that obtaining energy consumption
figures for real network infrastructures represents a very challenging task (due to the
inconsistency of the different models, which further become quickly out-of-date). We thus take
special care in the definition of a general model, describing the devices’ energy consumption as
a function of their utilization. The model is expressed in a parametric form, that makes it easily
extensible to other cases, and which we tune in this work according to power figures available
in [42].

![Graph of different models for network device energy consumption](image)

Figure 3.1 - The different models for the network device energy consumption,
expressed as parameterized function of the device utilization.

It is generally accepted that network device energy consumption grows linearly between a
minimum value $E_0$, which corresponds to the idle state, and a maximum value $M$, which
-corresponds to the maximum utilization [46]. Furthermore, a null energy consumption is
assumed when the device utilization is equal to 0, in which case the device is set to a sleeping
state. We refer to this model as “idleEnergy”, which is illustrated in Figure 3.1 by a solid line.
For what concerns the actual values of parameters $E_0$ and $M$, we rely on the mostly accepted
and diffused energy figures available in the literature. Table 3.1 summarizes the parameters we
used, where $C$ represents the node capacity. As the overall switching capability for nodes in the
considered topology is not available, we considered a node as being able to switch the double of
the sum of the capacity of all links connected to it. This is a design conservative choice, which
would allow the network manager to add a reasonable number of links without having to
change the devices.

Table 3.1 - Energy consumption parameters in Watts, for the different network elements

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>$0.85\sqrt[3]{C}/2$</td>
<td>$\sqrt[3]{C}/2$</td>
<td>[44]</td>
</tr>
<tr>
<td>(0-100] Mbps links</td>
<td>0.48</td>
<td>0.48</td>
<td>[45], [42]</td>
</tr>
<tr>
<td>(100-600] Mbps links</td>
<td>0.90</td>
<td>1.00</td>
<td>[45], [42]</td>
</tr>
<tr>
<td>(600-1000] Mbps links</td>
<td>1.70</td>
<td>2.00</td>
<td>[43]</td>
</tr>
</tbody>
</table>

Two special cases of this energy model are of particular interest in our analysis. In the fully
proportional model, the parameter $E_0$ is equal to 0. This model represents an ideal case where
energy consumption varies linearly with the device utilization, between 0 and $M$. This model is
illustrated in Figure 3.1 by a dashed line. It represents the behavior of fully energy-aware
devices, such as communication links supporting rate adaptation [47]. Nodes could also
present such a behavior when their components are regulated in function of the load (e.g.,
Dynamic Voltage Scaling (DVS), modular switching fabrics, etc.). The fully proportional model
is thus a resultant of several green technologies, which are not necessarily available today, and
is thus to be considered as a futuristic scenario. On the opposite, in the energy agnostic model
the $E_0$ parameter is equal to $M$, as illustrated in Figure 3.1 by a dotted line. This case models
network elements whose energy consumption is constant, independently from their load,
and are never powered down (i.e., the common case today).

3.3 Problem Formulation

We represent the network as a directed graph, $G=(N,L)$, with $N$ the set of nodes modeling
interconnection devices and $L$ the set of arcs modeling the communication links. For any
network element $a$ (node or link), we will denote by $l_a$ its load and by $c_a$ its capacity, i.e., the
maximum load it can support.

Our objective is to find the network configuration (i.e., the loads and the on/off status of the
nodes and links of the network) that minimizes the total network energy consumption,
expressed as the sum of the consumptions of all nodes and links. Considering the model
introduced in the previous section, the consumption of each element corresponds to an affine
function of the element usage, i.e., the ratio between its load and its capacity. The constant
term of this affine is equal to $E_0$ when the element is switched on and is null otherwise. To
model this, we denote by the binary variable $x_a$ the status of element $a$ ($x_a=1$ whenever $a$ is on
and $x_a=0$ otherwise). The slope of the affine function corresponding to element $a$ is denoted
$E_{fa}$. Finally, links are full duplex and they are considered entirely powered as soon as one
direction conveys traffic. Since in the above graph formulation the two directions are
separately modeled, the link load is the sum of both directions loads. With this model, the
network total energy consumption may be represented by the following expression (where the first sum needs to be divided by a factor 2 in order to avoid counting links twice):

$$\frac{1}{2} \sum_{(i,j) \in L} \left( \frac{l_{ij} + l_{ji}}{c_{ij}} E_{ij} + x_{ij} E_{0ij} \right) + \sum_{n \in N} \left( \frac{l_{n} E_{in}}{c_{n}} + x_{n} E_{0n} \right)$$  (1)

The load imposed to this network is defined by a traffic matrix that specifies, for every couple of ingress and egress nodes \((s, d)\), the traffic flowing from \(s\) to \(d\), denoted by \(r_{sd}\) hereafter. This flow from \(s\) to \(d\) is routed across the network, generating a traffic of \(f_{sd, ij}\) over any link \((i,j)\), subject to the usual flow conservation constraints.

As mentioned above, to preserve QoS, no links should reach a 100% utilization, or more in general, an arbitrary value \(\alpha\) that the network operator considers safe enough. This defines the following set of constraints:

$$\sum_{(s, d) \in N^2} f_{sd, ij} = l_{ij} \leq \alpha c_{ij} \quad \forall (i,j) \in L$$  (2)

We further assume node load to be directly proportional to the traffic entering and leaving the node. In particular, we consider that they are equal, which adds the following constraints to our problem:

$$l_{n} = \sum_{(i,n) \in L} l_{in} + \sum_{(n,i) \in L} l_{ni} \quad \forall n \in N$$  (3)

Finally, we consider that a node or a link is switched off as soon as its load is equal to zero. This allows to relate variables \(x_{a}\) and \(l_{a}\) for any element of the network through the following sets of constraints:

$$Z x_{ij} \geq l_{ij} \quad \forall i,j \in L$$  (4)

$$Z x_{n} \geq l_{n} \quad \forall n \in N$$  (5)

where \(Z\) is a “big” number (i.e., greater than twice the maximum between the nodes and the links capacities), used to force the variable \(x_{a}\) to take the value 1 when \(a\) has a load greater than 0, and the value 0 when \(l_{a} = 0\).

Minimizing the total energy consumption (1) while satisfying all the constraints mentioned in this section is a mixed integer program, with binary variables \((x_{a})\) and continuous variables \((l_{a})\).
3.4 Experimental Results

A known problem in the evaluation of energy saving solutions is the lack of standard conditions and metrics. At the same time, a major concern in the green IT field is the ability to quantify the achievable energy reduction in a scenario that is as relevant and as objective as possible – as otherwise the promised energy gain could be as extraordinary as, unfortunately, highly unrealistic.

For this reason we decided to evaluate the potential benefits of energy-aware routing on solution using realistic, and publicly available, data. The setting we choose constitutes a worst-case scenario, so that we are able to estimate a reliable and conservative lower-bound on the achievable gain (first subsection). On such realistic scenario, we also elaborate on further properties of the solution, so to assess the impact of energy-aware routing on the achievable quality of service, possibly imposing maximum load on individual links to ensure further robustness (second subsection). Finally, as we are aware that worst-case scenario may provide a too pessimistic lower bound, we perform a careful sensitivity analysis of the solution, widening the boundaries of our investigation through careful transformation of the real input traffic matrix (third subsection).

3.4.1 Worst-case scenario

As input of our reality-check, we selected the GEANT topology [48], which represents a real and fairly complex network: as Figure 3.2 depicts, it includes 23 nodes and 74 links. As traffic data, we selected a subset of the available Traffic Matrices (TMs), specifically 24 TMs, taken at hourly intervals between 00:30 and 23:30 of 5/5/2005. Notice that this TM set includes the complete traffic variations of a standard working day.

Figure 3.2 - A representation of the GEANT network topology used in the solution evaluation (different link colors represent different utilization levels)
As a performance metric, we selected the percentage of energy saved with respect to a routing configuration using the IGP Weight-Optimization (IGP-WO) algorithm [49]. IGP-WO is the standard practice in the operator networks; we will refer to this reference scenario as “IGP-WO routing”.

We model the optimization problem with AMPL [50], and use [51] for its numerical solution. Detailed results obtained for the three considered energy models are summarized in Table 3.2, while Figure 3.3 offers a graphical view of the energy saving, separately considering the node and link contributions. Both Table 3.2 and Figure 3.3 report values averaged over the 24 TMs set.

### Figure 3.3 - Energy consumption in Watts for the different routing algorithms, applied to the different energy models.

### Table 3.2 - Energy consumption in Watts for different routing algorithms, applied to different power models and scenarios, averaged over the full set of traffic matrices. For the case of Green routing, the percentage of energy saving with respect to the corresponding IGP-WO case is reported in parentheses.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>IGP-WO routing</th>
<th>Green routing</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nodes</td>
<td>Links</td>
<td>Total</td>
<td>Nodes</td>
<td>Links</td>
<td>Total</td>
</tr>
<tr>
<td>Energy-agnostic</td>
<td>7676.00</td>
<td>59.12</td>
<td>7735.12</td>
<td>7676.00</td>
<td>-0.0%</td>
<td>7735.12</td>
</tr>
<tr>
<td>Idle Energy</td>
<td>6565.95</td>
<td>46.23</td>
<td>6612.18</td>
<td>6569.22</td>
<td>+0.05%</td>
<td>6599.56</td>
</tr>
<tr>
<td>fully prop.</td>
<td>307.21</td>
<td>10.97</td>
<td>318.18</td>
<td>286.69</td>
<td>-6.7%</td>
<td>291.79</td>
</tr>
</tbody>
</table>

Regarding the idleEnergy model, we can expect energy savings to be mainly a consequence of switching off network elements, since this avoids the idle energy consumption $E_0$. Indeed, it is clear from the values considered in Table 3.1 that the impact of the fixed component $E_0$ on
the overall network energy consumption is much greater than the proportional energy component due to the device load ($M-E_0$). Moreover, in the considered model, the energy parameters of the nodes are generally two orders of magnitude larger than the ones of the links. This means that the energy saving achievable by switching off links represents a small contribution to the total energy saving. However, given the topology and the traffic level, it is generally not possible to switch off nodes (since every node is source and destination of traffic requests), but it is possible to switch off links. The GEANT scenario represents hence a worst-case scenario for our solution, lower bounding the achievable energy saving.

![Figure 3.4 - Overall network energy consumption under the idleEnergy model, and IGP-WO and green routing cases.](image)

The above considerations are verified in the results, which show a small energy saving due to nodes but a considerable one due to links, summing to a modest overall energy saving (about 0.2%). Figure 3.4 shows how the network energy consumption varies for the idleEnergy model when considering the different traffic requests on a typical working day (i.e., over 24 hourly traffic matrices). Energy consumptions are reported for both the IGP-WO and green routing, along with the energy saving percentage.

In the case of the fully proportional model, the energy saving is a consequence of the aggregation of traffic over paths involving the most energy efficient devices, while we are not interested in switching off nodes and links since there is no idle energy consumption ($E_0=0$). Observing the results reported in Table 3.2, we can see that it is possible to achieve a much higher energy saving by means of energy-aware devices (fully proportional model) than with nowadays devices, presenting at most a partial energy awareness (idleEnergy model). Therefore, we see that green routing and green technologies (such as link rate adaptation in IEEE 802.3az [52] and dynamic voltage and frequency scaling [53] techniques, which bring links and devices close to a fully proportional model) naturally interact for enhanced saving performance.
3.4.2 QoS Considerations

In our solution, energy saving comes as consequence of switching off network elements and optimizing their utilization level with respect to their power consumption. This strategy is in opposition to the common practice to guarantee robustness and QoS in networks: i.e., redundancy of network elements and distribution of the charge over all the available paths. It is therefore imperative to analyze the variations in the network device load that energy-aware routing brings with respect to the standard IGP-WO routing case.

![Link load distribution under IGP-WO and green routing.](image)

Figure 3.5 - Link load distribution under IGP-WO and green routing.

More precisely, we assess how a green solution displaces the load in the network to achieve energy saving, and how it affects the devices load, a performance indicator that has direct influence on users QoS. For the sake of simplicity, we report results that refer to a single scenario (namely, the 00:30 TM under the idleEnergy model), although qualitatively similar considerations hold for other scenarios as well. Figure 3.5 reports the distribution of the link loads for idleEnergy model for both IGP-WO and green routing. Note that in the IGP-WO case none of the link are idle, while our solution brings a considerable number of link to a zero utilization (i.e., since those links are turned off to save energy). As a consequence, green routing also increases the number of links with a higher utilization level, since this is a straight consequence of aggregating the traffic on a subset of the network devices, to be able to switch off the others.

Figure 3.6 shows the average link load for the IGP-WO and green routing cases: we can see that the overall average load slightly increases under green routing, and that our solution tends to move the load from the “average capacity” links, to the more energy-efficient “high capacity” links. Notice that resource consolidation is generally not possible for a number of (even less energy-efficient) “low capacity” access links, as they are located in more constrained areas of the network (i.e., at the edge, where the path diversity is lower). Notice also that the green solution does not increase considerably the average link utilization: indeed, even though resource consolidation may slightly increase the overall network load (as a consequence of longer paths), however this very same amount of traffic is generally shifted over a higher capacity link (with a thus limited impact on the link utilization).
Finally, it is interesting to further dig the solution of the optimization problem at a finer level of detail, by pinpointing individual links that are switched off under green routing. This is shown in Figure 3.7, where links represented as thick black lines are switched off. In the considered scenario, the switch off procedure only involves lightly loaded links: the average load on the switched off links is 5.2%. Moreover, only average and high capacity links are switched off (i.e., all link with a $C \leq 100$ Mbps are on). Overall, as all nodes connected to a switched-off link are also connected to at least another link with at least the same capacity, we can conclude that the impact of green routing on QoS will be minimal (as the traffic will be redirected on alternative links without considerably affecting the link utilization).

Nowadays, the network operators adopt as common practice to limit the load of the links to enforce QoS and robustness in their networks. In order to reflect this practice in our solution
and to obtain more realistic results, we introduce a maximum imposable load level for all the network elements, by a parameter referred to as $\alpha$. Figure 3.8 illustrates the variation of the achievable energy saving for a range of maximum imposable load $\alpha$, where in this case we average results over the full set of 24 TMs (results are obtained for the fully proportional model, but considerations hold for other energy models as well). From Figure 3.8 we gather that the reduction of the maximum device load does not significantly affect the achieved energy saving. This is due to the fact that the main limitation to the energy saving is represented by the topology and the traffic requests, rather than by the maximum device utilization (remember that, by design, nodes are never loaded more than 50%).

On the other hand, reduction of the maximum imposable load on links significantly affects the feasibility of the problem: it should be noticed that already in low-load scenario of 00:30 TM reported in Figure 3.5, some links are loaded more than 90% under IGP-WO. Hence, reduction of the maximum device load may easily result in unfeasible solutions (the percentage of feasible solutions is reported on the right y-axis of Figure 3.8).

![Figure 3.8 - Energy saving percentage against maximum imposable link utilization.](image)

### 3.4.3 Sensitivity Analysis

Finally, we perform a sensitivity analysis of the solution by performing another set of experiments, carefully controlling the evaluation scenario.

In this case, we consider that one or more nodes of the GEANT network are no longer generating and receiving traffic, but become “core” nodes which merely route other nodes traffic. As core nodes, we select the five most central nodes ($at1.at$, $ch1.ch$, $de1.de$, $es1.es$, and $uk1.uk$), and perform the full set of experiments by considering all combinations of $N$ core nodes, from $N=1$ (i.e., 5 scenarios with a single core node) to $N=5$ (i.e., the single scenario with all 5 core nodes). Clearly, in this case, the optimization solution may have the chance of switching off one or more of the core node, provided that routing is feasible under the reduced graph.

The aim of this set of experiments is twofold: on the one hand, we want to complement our worst-case analysis with other figures, which are at the same time realistic but controlled. On
the other hand, this kind of analysis is also important as it can bring useful insights for topology design.

![Graph showing energy savings](image)

**Figure 3.9 - Energy saving as a function of the number of core nodes.**

Results are reported in Figure 3.9 for the idleEnergy model and the 09:30 TM, which correspond to the minimum network load but not to the maximum of energy saving (as can be seen in Figure 3.4). Notice that, as the load is low, and nodes could be entirely switched off, there are opportunities for a higher energy saving: thus, the least loaded TM upper-bounds the achievable optimization gain.

Already when N=1 the total gain under the idleEnergy model is about 6% – corresponding to a 30-fold increase with respect to the 0.2% gain early obtained by switching off links only. Clearly, 6% roughly correspond to switching off 1/23 nodes (notice that node consumption is not homogeneous). As it can be seen in Figure 3.9, this trend does not hold for growing number of core nodes, since the optimization problem is not always able to switch off all the core nodes under the routing constraints. In fact, the blue line in Figure 3.9 reports on the right y-axis the average number of switched off nodes, which grows less than linearly with respect to the number of core nodes, due to the saturation of the switching capability of the network. In the same picture, the green line reports on the left y-axis a lower bound to the power consumption, obtained by switching off all the nodes which are of core type in the scenario.

Overall, we see that even though the network redundancy is high, energy-aware routing is effective in consolidating the unnecessary resources: thus, network topology could be dimensioned to avoid over-load for peak scenarios, and energy-aware routing could profitably be used to dynamically select the subset of necessary resources when in under-load scenario.

### 3.5 Conclusion

In this work, we study energy-aware routing as an optimization problem, evaluating its effects on the network energy consumption and on the network device load, a standard indicator for the QoS performance.

Considering the energy performance, the obtained results allow to better understand the mechanisms enabling energy saving, i.e., how the traffic is redirected to allow network devices to be switched off. Moreover, our results show that energy saving performance strongly depends on both (i) the network topology and traffic conditions and (ii) the device technology, corresponding to different power models. Indeed, green routing may provide an energy-
efficient automatic adaptation of the network resources to the traffic conditions, but this should be supported by the topology design (e.g., enough path diversity) and supported by energy-efficient device design.

Considering the network QoS performance, numerical results show that, at least in the considered scenarios, achieving energy saving does not necessarily negatively affect the network performance, even if it may raise reliability issues. Instead, results also show that imposing maximum load on links may significantly limit the applicability of energy-aware routing, as many solutions may become unfeasible. Thus, ISP will have to carefully select the trade off between the achievable energy efficiency gain and the robustness of the solution – as the choice of an unlucky robustness threshold may severally limit the achievable energy efficiency gains.

To the best of our knowledge, this work is the first to bring a careful and thorough reality-check on energy aware routing, by considering a publicly available real network topology, along with several traffic matrices and energy models; as a beneficial side effect, using publicly available data also promote cross comparison work. In future work, we aim at benchmarking existing heuristics such as [41] against the optimal numerical solution – both from the point of view of the achievable energy saving, and from the one of the robustness and QoS. We are also interested in further refining the energy profiles, in particular considering the underlying optical equipments. We believe that results may change significantly when taking into account optical links over the real geographical distances, as the energy consumption of long haul links would actually depend on their length (e.g., due to periodical signal regeneration).
4 LOCARN

LOCARN is an imaginative network paradigm compliant with many autonomic networking properties. LOCARN stands for "Low Opex and Capex Architecture for Resilient Networks". It aims to explore two concepts (auto-forwarding and enhanced broadcast) in order to increase as much as possible network simplicity and hence increase savings in both OPEX and CAPEX domains.

Main concepts of LOCARN may be successfully compared to the principles defined and specified for Ad-Hoc Mobile networks (e.g. Dynamic Source Routing, IETF RFC-4728). LOCARN addresses more specifically Transmission Networks by an intrinsic simplicity and the willingness to take into account some "Carrier Grade" properties.

As detailed later, LOCARN will be compliant with main properties encountered in Autonomic Networking (self-* properties).

This document aims to offer a sufficient enough in-depth view of the LOCARN network concepts with the objective to prepare its Proof-Of-Concept implementation. That latter will be done through TIGER2-WP5 activity.

4.1 Problem Statement

A LOCARN network is intrinsically able to auto-configure itself dynamically, taking into account the load of nodes and links within a rather low time scale compatible with the creation of a new service and/or any topological changes on the network.

To that end, a constant self-analyzing process allows that very quick (re) configuration of the network behavior. The bottom line of these operations that are "transparent" at the management level, is, from a service point of view, a network somehow "Plug&Play".

Moreover, in classical packet networks (e.g. IP or Ethernet), nodes use information included in the well-known "routing/forwarding tables" to forward data frames. The difficulty is to maintain up-to-date the content of these tables which implies the use of a dedicated control and/or management plane to fill in these tables. To alleviate this drawback, auto-forwarding paradigm (implemented in LOCARN) leverages of only information directly present in the header of the frame to switch it without the necessity to perform look-up in any table in each transfer node.

Obviously, the downside of this method is the need, at the Origin edge port, of a specific control plane to build the routing information added in the frame header. The global principle proposed in LOCARN relies on the use of a broadcast to discover the possible paths once a given service is under creation (pre-registered at the management plane).

Several variants of that principle may be conceived and implemented in LOCARN with different performance objectives. However, all should use specific control frames as PATH_REQUEST/PATH_DISCOVER to elaborate GO_PATH and BACK_PATH information to be inserted in each frame header at the Origin Edge Ports.

In this document, a rather basic broadcast named "enhanced incremental broadcast (EIB)" is used and specified.
4.2 Solution

This paragraph proposes a generic description of LOCARN components. Frames used in a LOCARN network are listed and described. Then the structure of a possible LOCARN node is specified both at the architecture and functional levels. Finally, some other issues will be introduced with the willingness to open possible further studies.

4.2.1 Frame

The instantiation of LOCARN paradigm as a packet transport network (as targeted in TIGER2 project) leads to cover three main functionalities:

- the transport itself of data frames
- the service creation between Service End points (discovery process)
- the service maintenance in case of any change in server layers as an example (e.g. topology change, ...)

In order to perform these functionalities, several kinds of LOCARN frames are specified. Only five types of frames are necessary to cover these three items and to make the LOCARN solution an operational network.

Each LOCARN frame is referenced by a LFT (LOCARN Frame Type). Details regarding LOCARN frame format structures are provided in Annex A: Generic LOCARN Frame Formats.

Let’s notice that this section specifies generic LOCARN frames that are agnostic to any server layer. Any real LOCARN implementation will need to adapt LOCARN frames to a specific server layer.

4.2.1.1 Frame dedicated to the Transport function

The LOCARN Data Frame (LDF) is the internal frame format inside the LOCARN network. It is composed of the LOCARN payload (DF) and the LOCARN overhead.

a) The LOCARN payload is somehow the "customer" frame entering inside the LOCARN network and named here Data Frame (DF). DF is transparently routed from a physical Edge ingress port to a physical Edge egress port.

b) The LOCARN overhead contents the information necessary for the DF propagation along the LOCARN path (from one Edge to another one). Among other things, the LOCARN overhead contains the LOCARN Service Identifier.

4.2.1.2 Frames dedicated to the Path Discovery Process

The path discovery process uses two kinds of frames:

a) The Path Request LOCARN Frame, generated by a Service Origin Edge Point and broadcasted inside the LOCARN network. To provide the Enhanced Incremental Broadcasting (EIB) process, each node adds some information to an incoming Path Request before relaying it. This information mainly contents some identifiers (input port, output port and optionally the node identifier) and some information about the estimated "quality" to relay the frame through this port on this node. Other more classical information items are generated by the Service Edge Origin Point like LOCARN Frame Type and the Service Identifier.

b) Path Discover LOCARN Frame is generated by a Service Edge Destination Point upon reception of a LOCARN Path Request if the Service Identifier contained in the
LOCARN Path Request received has been previously "registered" at this destination point. This destination point uses the Information inside the received Path Request frame to compute the path to go back to the Service Edge Origin Point and transmit a copy of the information found in the received Path Request LOCARN Frame to inform it that a path exists from the Service Edge Origin Point to the Service Edge Destination Point.

4.2.1.3 Frames dedicated to the Service Maintenance

Two types of frames are related to the Service maintenance process:

a) The Hello Forward LOCARN Frame initiated by the Service Edge Origin Point at periodic interval. This frame is carried in the Data Plane using the standard LOCARN auto-forwarding scheme.

b) The Hello Back LOCARN Frame initiated at the Service Edge Destination Point upon reception of an Hello Forward LOCARN Frame.

Note that these two types of frames carry a Service Identifier. Thus, the maintenance process is related to each End-to-End service.

4.2.2 Node

A LOCARN Node may be described in a "mecano" way as an organization of different functional blocks. This way, a LOCARN node is composed of the three following functional blocks as depicted in Figure 4.1:

- LOCARN Matrix Functional Block (LMFB)
- LOCARN Tee Path Request Functional Block (LTPRFB)
- LOCARN Port Functional Blocks (LPFB). That latter can also be seen as a compound block and divided into two sub-blocks:
  o The LOCARN Port Edge Origin Functional Blocks (LPEOFB)
  o The LOCARN Port Edge Destination Functional Blocks (LPEDFB)

![Figure 4.1 - LOCARN Node (4 ports)](image-url)
LMFB is common for the whole node whereas LPFB and LTPRFB are local to the physical port.

The role of each bloc is described below.

### 4.2.2.1 LOCARN Matrix Functional Bloc (LMFB)

The LOCARN Matrix Functional Block (LMFB) is only able to manage LOCARN Data Frames. If the frame is not recognized as a LOCARN Data Frame (role of LFT), the frame is discarded. Else, LMFB handles the LOCARN Data Frame header in order to relay it to the relevant egress port according to the following actions:

- Look-up inside the LOCARN Data Frame header the value of the pointer LOCARN Path Hop (LPH) indicating the current position (i.e. the current node) in the whole path.
- Select the egress port given by the pointer LPH
- Increment LPH (will be used by the following nodes)
- For security reason, if the new value of LPH is greater than the LOCARN Path Length (LPL) contained in the LOCARN Data Frame header, discard the frame
- Push the frame (modified with LPH incremented) to the relevant port.

### 4.2.2.2 LOCARN Port Edge Origin Functional Bloc (LPEOFB)

As mentioned above, the LOCARN Port Functional Blocs (LPFB) is divided into two LPEOFB and LPEDFB sub-blocs. Their internal structure is depicted in Figure 4.2.

![Figure 4.2 - Interconnection of sub blocs in a LOCARN port](image-url)
The LPEOFB sub-bloc performs the following actions:

**ClFT (Classifier Frame Type):**
- Classify frames to relay them to the relevant block
  - All LOCARN Frames, except Path Request LOCARN Frame, towards the LMFB
  - Path Request LOCARN Frame towards the PrMark
  - Other frames towards the PortEO to be eventually transformed into LOCARN Data Frame.

**PortEO:** Manage the Service Origin Edge Point for this physical port:
- Generate Path Request LOCARN Frame for services in case of service creation or service maintenance
- Generate Hello Forward LOCARN Frame for all active services
- Encapsulate the Data Frames adding the LOCARN overhead

**PrMark:**
- Temporary mark the frame with the Port Id (creation of an Internal Path Request Frame - IPR)
- Push the IPR to the LTPRFB associated to this port

Note that the use of IPR is only to manage (store) the input port Id during the transfer of the Path Request LOCARN Frame inside the node. IPR will be transformed into a Path Request LOCARN Frame later at the egress port once the output port Id will be known and added.

### 4.2.2.3 LOCARN Port Edge Destination Functional Block (LPEDFB)

The LPEDFB sub-block performs the following actions:

**ClLocal (Classifier Local):**
- Classify the frame according the LPH and LPL values to determine if the frame must be parsed by PortED or directly pushed downside (towards OAD).

**ClassOut:** Used in the case the frame must be handled locally. In this case, classify the frames between
- Path Discover LOCARN Frame and Hello Back LOCARN frame transmitted to the PortEO
- Hello Forward LOCARN Frame transmitted to PortED

**PortED:** Manage the Service Destination Edge Point for this physical port:
- Generate Path Discover LOCARN Frame upon reception of an Internal Path Request (IPR) if the Service Identifier has been registered on this port. The Path Discover LOCARN Frame includes information from ingress and egress ports Id and Qlo (Local Quality measurement of the bit-rate of the link connected to this port as well as an estimation of the used rate for this link (queue filling rate, rate measurement, ...)
- Generate Hello Back LOCARN Frame upon reception of Hello Forward LOCARN Frame
- De encapsulate the LOCARN Data Frames to retrieve the customer Data Frame

**OAD:**
- Output Adapter (Queuing, shaping...)

---

TIGER2 – Together IP, GMPLS and Ethernet Reconsidered, Phase 2
4.2.2.4 LOCARN Tee Path Request Functional Bloc (LTPRFB)

Each LFTFB duplicates IPR and pushes the cloned frames to all LPEDFB of the node (all other ports). No change is made inside the frames.

4.2.3 LOCARN Frame processing

This section analyses the relation between LOCARN Frames and functional blocs described above.

**PR** (Path Request LOCARN Frame)
- Pre generated by PortEO
- Finalized and transmitted by PortED

**HF** (Hello Forward LOCARN Frame)
- Generated by PortEO
- Upon reception, PortED generates an Hello Back LOCARN Frame

**LDF** (LOCARN Data Frame)
- Generated by PortEO upon reception of a DF
- De encapsulated by PortED
- Routed by LMFB

**DF** (Data Frame)
- Customer frame entering at the boundary of the LOCARN Network
- Encapsulated or discarded by PortEO
- De encapsulated by PortED

**PD** (Path Discover LOCARN Frame)
- Generated by PortED
- Analyzed and discarded by PortEO

**HB** (Hello Back LOCARN Frame)
- Generated by PortED

**IPR** (Internal Path Request)
- Generated by PortEO
- Duplicated by LTPRFB
- Analyzed and completed by all PortEDs

4.2.4 Other issues

This section aims at introducing some open issues that have not been studied yet but that should require more in-depth investigations. It can not be considered as part of the LOCARN specification itself.

4.2.4.1 Specific Path Request and Path Discover issue (functional issue)

At the path discovery step, information about nodes (more exactly about ports) are collected by the Path Request LOCARN Frame and returned at the Service Edge Origin Point using the Path Discover LOCARN Frame. At the generation step of the Path Discover LOCARN frame, specific information about the service itself (cost for example) may be added by any potential Service Edge Destination Point.

This information may be used by the Service Edge Origin Point to choose "the best" Service Edge Destination Point and "the best" way to be connected to this Service Edge Destination point.
The exact semantic of "THE BEST" depends on the kind of service managed by the LOCARN network. Any LOCARN implementation MUST instantiate:

- Specific fields to carry the information in the Path Request LOCARN Frame
- A specific algorithm to compare Path Discover LOCARN frame content.

### 4.2.4.2 LOCARN Edge Port specificities (architecture issue)

This item raises the question to declare or not ports at the boundaries of the LOCARN network as LOCARN Edge Ports.

Beyond the question of the security, this LOCARN Edge Port specificity brings network architecture interests and scalability improvements. Indeed, if a port is stamped as LOCARN Edge Port, all incoming frames will be processed as DF, even if they are already LDF. By this way, we can manage an overlay "LOCARNinLOCARN" network architecture and increase the scalability with a LOCARN hierarchy.

### 4.2.4.3 "Big" node architecture (scalability issue)

Inside LOCARN Frames, the LOCARN Path (pointed by LPH) is a part of the traffic overhead inside the network. This overhead is linked to the network size. For rather big network, it could be interesting to reduce this extra-traffic part generated by this overhead by reducing both a) the size needed to code each port, b) the network diameter accessible by a node.

So, it is important to avoid big Port Identifier fields. The drawback is clearly to limit the size of the node (the number of ports). A workaround may be to have a specific node architecture where "virtual LOCARN nodes" are interconnected together by a LOCARN port (Virtual LOCARN nodes have exactly the same specification as LOCARN nodes). In this case, the "Big" LOCARN node will have to handle the LPH as many time as the number of Virtual LOCARN nodes crossed over.

This mechanism offers the ability to bypass the limitation of number of ports on a LOCARN node. The drawback is a reduction of the networks diameter that includes all LOCARN nodes, virtual or not.

### 4.2.4.4 "LOCARN Path Port List" size (interoperability issue)

In order to increase the Plug&Play property of a LOCARN network, it may be interesting to use equipment with different LPL (LOCARN Path Length) and to expect that the network take automatically the different values of LPL acceptable by the equipment of the network. With such this very useful enhancement of the protocol, some changes have to be made in the LOCARN Frames management.

At best, an enhancement of the Path Request / Path Discovery protocol will be able to automatically discover the LPL per service.

### 4.3 Preliminary evaluation results

#### 4.3.1 LOCARN PoC implementation

A first LOCARN implementation has been realised using this specification and demonstrated in the scope of TIGER2 – WP5 work package activity.
4.3.2 LOCARN compliancy with Autonomic properties

LOCARN intrinsically offers autonomic properties, simplicity and efficiency. Most of these autonomic properties are directly linked to the frame management and fundamentals of LOCARN. As an example,

- self-discovery (Implicit)
  using Path Request LOCARN frame and Path Discover LOCARN Frame
- self-configuration (Implicit)
  discovered information about the network are automatically usable by the Data Plane. Only Services need to be configured
- self-healing
  End-to-end maintenance OAM frames per service
- self-optimization
  Periodic Path Request LOCARN Frame can help to reconfigure the path of services taking into account the availability of network resources
- self-management (not applicable, not needed)
- self-protection
  Possibility to define Edge Ports with filtering behaviors. Only "registered" services are allowed on the network
5 Adaptive Probabilistic Flooding for Multipath Routing in LOCARN

In this chapter, we focus on an algorithm to improve the performance of the previously presented LOCARN architecture. As a recall, LOCARN relies on the following key principles:

- each source maintains a set of multiple distinct paths to each destination;
- the header of each packet consists of a sequence of labels, one per node on the corresponding selected path;
- each node receiving a packet (i) removes the first label of the header, if any, and forwards the packet accordingly (the node is a relay) (ii) sends it to the IP layer in the absence of label (the node is the destination).

Note that core nodes do not maintain routing tables, which avoid expensive address lookup algorithms. All routing information is contained in the packet itself. Paths are built and updated by the source nodes, which enables multipath routing. For instance, each source maintains a primary path to each destination (typically, the shortest path), as well as a secondary path in case of failure or traffic surge. Unlike traditional IP routing, this secondary path is always available, which enables fast restoration in case of failure.

A critical component of this architecture is the algorithm used to discover paths. We propose a novel flooding algorithm inspired by the opportunistic algorithms of peer-to-peer applications, which consists in adapting the flooding rate of core nodes to the number of already received discovery messages. We refer to this algorithm as adaptive probabilistic flooding. At the cost of a limited number of state variables in core routers, the algorithm is able to discover multiple paths in a quasi-optimal way. Using analytical bounds, we prove that, unlike pure flooding, the algorithm scales with the network size.

After related work is presented in the next section, the rest of the chapter focuses on describing the proposed path discovery algorithm and providing a detailed analysis of its performance.

5.1 Related work

Routing is a critical component of the Internet, and as such has long been studied by the scientific community. The problem of finding a single path interconnecting any two nodes of a graph is solved by well-known algorithms like Dijkstra and Bellman-Ford, which have been implemented in widely deployed protocols such as OSPF and RIP, respectively. However, interconnecting nodes through a single path (typically, the shortest) does not make the network resilient against failures and traffic surges. Hence, different techniques relying on multiple paths have been proposed. For instance, ECMP [54] aims at balancing load over multiple paths of equal cost. In standard IP/MPLS networks, the control and data planes are generally considered jointly; multipath routing is then achieved through a centralized algorithm, solving some standard multi-commodity flow problem [55, 56].

In this work, we focus on the control plane and address the issue of the efficient discovery of multiple paths, as in [57-64]. Some of the above papers consider the problem of finding a pair of disjoint paths between any two nodes within the network, considering that the primary (shortest) path is already known. In [57-59], the problem is shown to boil down to a standard shortest path problem by some appropriate modification of the original graph. Ogier, Rutenburg and Shacham define in [57] an algorithm that achieves such a graph modification
and evaluates its performance in terms of communication, convergence time, and space complexity. Sidhu, Nair and Abdallah enhance this algorithm in [59] by finding all possible disjoint paths between any two nodes of the network. Eppstein proposes in [58] an algorithm that finds the first $k$ shortest paths between any two nodes by a breadth first search of a 4-heap in which every node represents a path. Finally, works like [60] take a more practical approach, and enhance IP routing by means of centralized algorithms to determine disjoint paths, and distributing such paths through the routers taking care of avoiding loops.

The problem of finding multiple paths without any a priori knowledge such as the shortest path is quite different. Most algorithms then rely on flooding [61-64]. A swarm intelligence based solution is proposed with the Ant Colony Optimization (ACO) algorithm [62], where a set of ants is spread through the network in order to discover disjoint multiple paths: the pheromone left by the ants is employed in order to avoid already crossed paths. In [63], a flooding algorithm on layered routing architecture is employed: the basic idea is to give a score to each packet and to decrease this score for each link on which the packet is flooded; only nodes along the best path can increase the score and re-flood the packet. Authors in [64] propose a strategy where a scout message walks through the network accumulating nodes discovered in the walk: nodes re-flood the message only when the current discovered path differs significantly from the stored shortest path, allowing thereby to find multiple paths. In a very different context, namely ad-hoc wireless networks, flooding-based technique are exploited by Dynamic Source Routing (DSR) [61] where however multiple paths are not taken into account.

To the best of our knowledge, our approach introduces a number of novel ingredients. Similarly to DSR [61], in our approach the source inserts one label per node on the path to the destination, and the data plane forwards the packet by popping a label from the packet header at each hop. Unlike DSR, we don’t have collisions, or problems inherent to wireless networks, yielding to a radically different algorithm design. The adaptive probabilistic algorithm we propose greatly limits the number of exchanged messages, as in [63], without however requiring packets to carry the scores associated with the discovery algorithm. The limited amount of state kept by nodes is not used to avoid crossing already traveled paths as in [62], but rather to avoid taking these paths too often, which has important consequences on the quality of the multiple paths discovered. Finally, unlike [64], the algorithm does not rely on threshold-based decisions, nor it depends on topological properties of the network – rather, its design makes it robust and auto-terminating irrespectively of its actual parameter setting, with performance that degrades gracefully in case of parameter misguidance.

5.2 Path discovery algorithm

5.2.1 Overview

We aim at designing a distributed algorithm for path discovery, capable of finding multiple, possibly disjoint, paths between any pairs of nodes. To do so, each node periodically advertises its presence by means of some flooding procedure described below. Specifically, each node sends an advertisement message every $t_a$ seconds; typical values range from a few seconds to minutes [56]. Besides, each node sends keep-alive messages over each path in order to ensure that this path has not failed; the corresponding refresh messages are sent every $t_r$ seconds, typically set to a few milliseconds [65].
During the flooding procedure, each relay node adds its identifier to the advertisement messages it receives, so that these messages carry information concerning the whole traveled path. Upon reception of an advertisement message, a node learns a path from the source of this message, as well as from any intermediate node on this path, as in DSR [61]. Flooding decisions are taken independently by each node, and constitute the core of the algorithm. The main idea is that nodes need to flood a received message at least once, so that shortest paths are discovered. Nodes actually need to flood the message multiple times, in order to discover further paths beyond the shortest one. The number of flooding decisions is critical with respect to both the quality of the path discovery and the overhead of the algorithm.

A simple option could consist in including a Time To Leave (TTL) field in the packet, so as to interrupt the flooding process when some pre-configured maximum path length is reached. The selection of a proper TTL value is critical in this case: if the TTL is shorter than the graph diameter $D$, for instance, then connectivity cannot be guaranteed; if the TTL is too large, the overhead of the algorithm becomes prohibitive (as the number of relayed messages is exponential in the TTL).

We propose an alternative approach based on adaptive probabilistic flooding. Any node receiving some advertisement message from source $s$ floods this message the first time, and floods it with some decreasing probability the following times. Specifically, node $i$ floods an advertisement message generated by source node $s$ over all its links (except the one from which it has received the message) with probability:

$$P = b^{n_{i,s}}$$  \hspace{1cm} (1)

where $b$ is some fixed parameter and $n_{i,s}$ is a counter, stored at node $i$, of the number of times node $i$ has already received an advertisement originated by node $s$. The flooding decisions are taken independently on each link, and the counter is reset periodically, as explained later. Note that node $i$ floods the first advertisement message it receives for source node $s$ since $n_{i,s}=0$ in this case. As further messages are received, flooding will become exponentially less likely, according to the backoff parameter $b$. The quality of the path discovery is expected to increase with $b$, at the expense of larger overhead. However, we shall see that performance is not very sensitive to this parameter, which makes the algorithm robust and practically interesting.

### 5.2.2 Primary and secondary paths

Consider a network, modeled as an undirected graph $G=(E,V)$, composed of $|V|=N$ routers, in which any pair of adjacent routers are connected by a single link for simplicity (the algorithm can be easily extended to the general case of multiple links between any pair of nodes). Between any two routers $i,j \in V$, we are interested in finding a pair of paths, i.e., sequences of edges connecting node $i$ to $j$. We denote by $P$ and $S$ the primary and secondary paths, respectively, returned by the adaptive probabilistic algorithm on graph $G$, as described below. We denote by $L_p$ and $L_s$ the respective lengths of these paths.

To gauge the quality of the primary and secondary paths found by our algorithm, we need to define target path properties. The primary path is expected to be the shortest path in number of hops; in other words, we say that $P$ is optimal if it belongs to the set of shortest paths from $i$ to $j$ in $G$ (as there may be several such paths). The secondary path is expected to minimize the similarity with the primary path, $P \cap S$. Note that this choice reduces the share of faith between these paths, improving network resilience against failures and traffic surges.
To find the optimal secondary path $S$, we consider a modified graph $G'$ in which the cost of links along the primary path $P$ are increased by the network diameter [57], and other link costs are unitary. As links belonging to $P$ are now discarded due to higher cost, running Dijkstra on $G'$ we retrieve a path $S'$ minimizing the similarity function $P \cap S'$ (notice that since nodes along the primary path are not removed from $G'$, they can be included in $S'$ only if strictly necessary as the path would otherwise be disconnected). We say that the secondary path found by the algorithm $S$ is optimal if $|P \cap S| = |P \cap S'|$ and $L_S = L_{S'}$, i.e., the length $L_S$ of the secondary path is equal to the length $L_{S'}$ of the optimal $S'$ (as there may be multiple disjoint paths minimizing the similarity with the shortest path).

### 5.2.3 Metrics

To precisely quantify the overhead vs. path quality tradeoff early outlined, we resort to the following metrics. The average amount of messages handled by any given node during the advertisement procedure is denoted with $M_a$. For each path, the refresh process then requires each node to send keep-alive messages: we denote by $M_r$ the average number of such messages (counted once per every link traveled). The relative weight of $M_a/(M_r + M_a)$ quantifies the overhead induced by the advertisement process.

In the following, we evaluate the algorithm in terms of connectivity along the primary and secondary path (i.e., whether paths $P$ and $S$ joining any two nodes $i,j \in V$ exist) and optimality (i.e., whether $P$ and $S$ are optimal according to the above definitions). We express connectivity in terms of the probability $C_P$ (respectively, $C_S$) that, $\forall i,j \in V$, nodes $i$ and $j$ are connected by some primary (respectively, secondary) path. We express optimality in terms of the probability $O_P$ (respectively, $O_S$) that the primary path is also the shortest (respectively, that the secondary path is the shortest, most diverse path).

### 5.2.4 Pseudocode

A pseudocode description of the algorithm is given in Figure 5.1. A source node $s$ initiates the advertisement process by flooding an advertisement packet ADV to all its neighbors. The flooded packet contains a list of node identifiers $ID$, initially set to $ID[0] = s$ by the source, to which each node appends its own identifier. Upon reception of an advertisement packet ADV, a node learns a (backward) path to the source $s$ and to any intermediate node $d = ADV.ID[i]$ along the path. In case the receiver $j$ detects a loop (finding its identifier within the $ID$ list), it discards the message and aborts the flooding procedure. Otherwise, it analyzes, and possibly stores, the newly learned path $O_{j,d}$. Specifically, the primary (and secondary) path is first set if not existent yet. Also, if the newly overheard path is shorter than the primary path $length(C_{j,d}) < length(P_{j,d})$, then the primary path is updated with the overheard one. Similarly, if the overheard path has lower similarity than the current secondary path $|P_{j,d} \cap S_{j,d}| < |P_{j,d} \cap S_{j,d}|$, or if it has equal similarity but is shorter than the secondary $|P_{j,d} \cap S_{j,d}| = |P_{j,d} \cap S_{j,d}| \land length(C_{j,d}) < length(S_{j,d})$, then the secondary path is updated.
Figure 5.1 - pseudocode of LOCARN adaptive flooding procedure

Notice that we expect messages on the shortest path to reach a node before messages that take longer paths: in case of homogeneous setup (i.e., equal latencies) this invariant always holds. As such, the shortest path is chosen relatively early in the advertisement process and is typically never changed later on, as its change would also affect the quality of the secondary path. Not shown in the pseudocode for the sake of clarity, ties in the secondary path selection are broken at random.

Finally, after having added its own identifier to the ADV.ID, the node probabilistically floods the ADV message, with independent decisions per each neighbor (except the node ADV.ID[i-1] from which the message came), and update the per-source counter $n_s$. As we shall see, the duration of each discovery phase is much shorter than the discovery period $\tau_a$, which implies that the counter $n_s$ can be safely reset for new discoveries.
5.3 Performance evaluation

![Figure 5.2 - Tiger2: an example of hierarchical AS network. The core network is highlighted by the grey balloon, and router 0 represents the collect point toward the Internet core.](image)

In this section, we first evaluate the overhead of the algorithm through a simple analytical model, and then employ discrete event simulation to validate the analysis and evaluate performance in terms of the quality of discovered paths.

Simulations are carried on with Omnet++ [66] over four different network topologies, whose most significant properties are summarized in Table 5.1. Specifically, Table 5.1 reports the number of nodes $N$, the average and standard deviation of the node degree, $\bar{\delta}$, $\delta_0$ and $\sigma$, the average length of the optimal primary $L_p$ and secondary $L_s$ paths, the diameter $D$ of the original graph $G$ and the largest diameter $D'$ over the modified $G'$ graphs.

Note that we consider both real network topologies (Tiger2 [67], Abilene [68], Geant [69]), corresponding to different segments, as well as a set of 50 synthetic random graphs (Random) whose number of nodes and degree distribution loosely fit the real topologies (in case of the Random topology, $D$ and $D'$ are averaged over the 50 considered instances). All network topologies are well known except for Tiger2, which we depict in Figure 5.2. For the time being, we use homogeneous settings (i.e., constant and equal delay on every link), and no failures happen within the network. The evaluation of more complex (heterogeneous delay, failures, etc.) scenarios is part of our ongoing work.
Table 5.1 - Topological properties of the network scenarios

<table>
<thead>
<tr>
<th>Network</th>
<th>Segment</th>
<th>N</th>
<th>$\delta$, $\square$</th>
<th>$\sigma$</th>
<th>$\frac{L_p}{L_s}$</th>
<th>$D$</th>
<th>$D'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiger2 [67]</td>
<td>metro</td>
<td>22</td>
<td>3.6</td>
<td>0.6</td>
<td>2.7</td>
<td>3.7</td>
<td>5</td>
</tr>
<tr>
<td>Geant [68]</td>
<td>aggregation</td>
<td>22</td>
<td>3.4</td>
<td>1.4</td>
<td>2.6</td>
<td>4.0</td>
<td>6</td>
</tr>
<tr>
<td>Abilene [69]</td>
<td>core</td>
<td>11</td>
<td>2.6</td>
<td>0.5</td>
<td>2.4</td>
<td>4.2</td>
<td>5</td>
</tr>
<tr>
<td>Random</td>
<td>synthetic</td>
<td>22</td>
<td>3.2</td>
<td>1.6</td>
<td>2.6</td>
<td>3.8</td>
<td>5.4</td>
</tr>
</tbody>
</table>

(a) Number of advertisement messages generated by a full discovery round of all the nodes in the network as a function of $\beta$

(b) Number of refresh $M_r$ and advertisement $M_a$ messages generated during a fixed time window $W=1$ sec as a function of the advertisement period $\tau_a$ (Tiger2 network)
5.3.1 Overhead

Overhead generated by adaptive probabilistic flooding.

Pure flooding generates a number of messages that is exponential in the number of nodes $N$. Our adaptive probabilistic flooding algorithm is scalable in the sense that it generates only $O(N)$ messages, as shown below, at the expense of $O(N)$ counters, cf. (1).

Consider a single advertisement from some source node $s$, and consider some relay node $j$ with degree $\delta$. The first time node $j$ receives a ADV message originated by $s$, it sends a copy on each output link, except the one where the ADV message has been received. This generates $\delta - 1$ messages. The second time $j$ receives a ADV message from the same source, it will forward the message over each of the $\delta - 1$ links with probability $\beta < 1$; so, at second reception, node $j$ generates $(\delta-1)\beta$ messages on average. Iterating and taking into account $N$ advertisement processes (one per each node), we bound the total number of control messages $M_a$ that are seen by the average node:

$$M_a \leq N[(\delta, \delta-1)+((\delta, \delta-1)\beta+(\delta, \delta-1)\beta^2+\ldots)]$$

$$= N(\delta, \delta-1) \sum_{n=0}^{\infty} \beta^n$$

$$= N(\delta, \delta-1) \frac{1}{1-\beta}$$ (2)

(in other words, the whole network will carry $NM_a$ messages for a full round of $N$ advertisement processes). Note that (2) is an upper bound since we do not account for the detection of loops, which reduces the actual number of transmitted messages. While it is in principle possible to refine the bound by considering the probability that loops form, this can be done in closed form only for simple topologies such as random graphs [70]. Moreover, it turns out that this simple and conservative bound matches very well, as we will see, the empirical results found by simulation.
Note also that, as the first flood is always performed, convergence of the primary $P$ path to the shortest path is always guaranteed. Hence, the backoff parameter $\beta$ affects only the quality of the secondary path $S$: by tuning $\beta$, we can upper bound the algorithm overhead $M_a$ while matching the required level of path quality.

Now let us focus on the number of messages generated during the advertisement process. Figure 5.3(a) depicts, as a function of $\beta$, the upper bound (2) along with the number of messages $M_a$ gathered by means of simulation, for the four topologies early outlined. Note that the upper bound associated with Abilene network is computed using the fitted parameter values $(N, \delta, \Box) = (11, 2.6)$, while for the other 22-nodes networks, only the upper bound for the Tiger2 topology is shown $(N, \delta, \Box) = (22, 3.6)$ to avoid cluttering the picture. It can be noted that the overall number of messages generated by the advertisement procedure triggered by all nodes in the network remains low, topping to a few hundreds for high values of $\beta$. Also, notice that the number of messages generated in all the real networks is very similar, with Tiger2 as the worst case. This shows the robustness of our algorithm, given the various topological properties of the considered networks.

Although the overall number of messages remains low for any value of $\beta$, we miss a reference for comparison: more insights can be gathered from Figure 5.3(b), that compares the number of messages generated by the advertise procedure with the periodic keep alive messages due to the path refresh function. Considering a fixed observation time window of $W=1s$, we vary the $\tau_a$ period in $[0.1, 1]$ $s$ interval and set the refresh time to the typical value of $\tau_r=50$ ms. We then compute the overall number of refresh messages $M_r$ by taking into account the actual primary $P$ and secondary $S$ paths lengths gathered in simulation as $M_r = \frac{W}{\tau_r}(\text{length}(P)+\text{length}(S))$. We finally gauge the number of advertisement messages in simulation for the Tiger2 network, which represents the worst case, so as to gather conservative results. As the picture clearly shows, the overhead due to the advertise procedure is very low even for very small durations of the advertise period $\tau_a$ (well below 1sec), and even for large values of $\beta$: for $\tau_a \geq 1$ sec the overhead becomes clearly negligible for any $\beta \leq 0.9$.

Finally, Figure 5.3(c) depicts the evolution over (slotted) time of the number of messages carried over the Tiger2 network during an advertisement originated by a single source: we observe that, after some initial exponential growth due to the flooding process, the backoff factor kicks in and slows down the growth, which then dies out very fast, due to an increasing number of flooding paths being probabilistically cut out. This auto-termination feature is a very desirable property of the algorithm, and further suggests that advertisement periods do not need to overlap, but can rather be staggered. This could be achieved either with a simple policy (e.g., periodically at random within $[0, 2\tau_a]$) or with more sophisticated schemes (e.g., each node deciding autonomously whether to trigger a new advertisement depending on the measured control messages load). In turn, this implies that instead of keeping $O(N)$ counters (one for each source in case of advertisements in parallel), the system could perform advertisement in series and keep a small number of $O(1)$ counters (using a modulo function to solve unlikely contentions due to independent simultaneous triggering of advertisement processes by multiple nodes). This is an interesting direction for future research, which we aim at pursuing in the following.
Figure 5.4: Quality of the paths found by adaptive probabilistic flooding.

(a) Connectivity probability of primary and secondary paths

(b) Optimality probability of primary and secondary paths

(c) Average overlap between primary and secondary path $|\varrho \cap \Sigma|$ for non-optimal secondary paths overlap
5.3.2 Path quality

Let us now focus on the quality of the paths that the adaptive probabilistic flooding algorithm is able to find. For simplicity, we let each node advertise itself once at time $t=0$ and evaluate the connectivity and optimality of the primary and secondary paths. Results are averaged over 10 simulations over the real topologies, and over 50 graph instances in the synthetic Random graph case.

Figure 5.4(a) depicts the connectivity probability of the primary and secondary paths as a function of $\beta$: as expected, primary connectivity does not depend on $\beta$ and is always guaranteed. Since a primary path is always found, the connectivity index is relevant for the secondary path only: we see that all secondary paths are connected in all networks when $\beta \geq 0.7$ (which correspond to small overhead in Figure 5.4(b)).

Figure 5.4(b) reports the optimality probability of the primary and secondary paths as a function of $\beta$: again, since the shortest path is always eventually found, the optimality of the primary path is guaranteed. Thus, the optimality index is relevant only for the secondary path: we see that a significant percentage (from 60% to 85%, depending on the topology) of secondary paths is optimal even for a very low value of $\beta=0.1$, and that at least 90% of secondary paths are optimal for all considered topologies when $\beta \geq 0.8$. Moreover, we observe that optimality gracefully degrades $\beta$, and furthermore with similar (roughly linear) slope across all topologies. This is a desirable behavior: as no phase transition nor knee appear in the path quality slopes, tuning $\beta$ between low overhead (low $\beta$) vs. high path quality (high $\beta$) is not critical.

Finally, we dissect the reason behind the sub-optimality of some secondary paths. Recall that a secondary path is optimal if it is the shortest and most diverse path compared to the primary. Hence, sub-optimality of the secondary path may be due to either (i) a non-zero overlap between primary and secondary paths, $|P \cap S| > 0$, or (ii) a path with a stretch over the optimal secondary path larger than one $L_s/L_s' > 1$. Figure 5.4(c) depicts the overlap, i.e., the number of nodes that primary and secondary paths have in common, conditioning over the sub-optimal paths (i.e., the overlap of optimal secondary paths is not accounted for in the picture). As shown by the figure, sub-optimality seems to be tied to slightly more than one node in common as $|P \cap S| \in [1, 1.5]$. Furthermore, as the average overlap is always $|P \cap S| \geq 1$ for any $\beta$, we can conclude that overlapping paths are significantly more common that long-stretching paths.

5.4 Conclusions

We have presented a novel flooding based algorithm for multiple-path discovery: the algorithm trades a small amount of state in routers, i.e., $O(N)$ counters, in order to significantly limit the number of messages generated by flooding through an adaptive probabilistic algorithm.

Simple analytical bounds, confirmed by simulation results, show the overhead entailed by the advertisement procedure to be low (with respect to the amount of keep-alive messages needed to keep the path up-to-date) and auto-terminating (due to the multiplicative decrease of the flooding probability).

Simulation results also testify excellent performance in terms of path quality: connectivity and optimality of the primary path are achieved by design, while 90% of secondary paths are
also optimal when $\beta \geq 0.8$ (or otherwise decrease linearly for lower $\beta$). Interestingly, the low percentage of low path is due to a very limited amount of share of faith between paths (1.5 nodes in the worst case, for $\beta=0.1$).

As part of our future work, we want to carry on more realistic experiments on a wider set of topologies and further reduce the amount of state to $O(1)$. 

6 Self-optimization of network resource allocation in MSTP

In this study case there is a centralized entity in the network that is in charge of routing and network resource allocation, based on the knowledge of the actual network state. For each new traffic demand, the online routing allocates the necessary network resources, but it usually results in suboptimal allocation.

Self-optimization of network resource allocation aims to provide automatically without human intervention an optimal allocation (the minimum). Periodically an optimization model is run, then the calculated optimal resource allocation is compared with the actual one, and if their difference is large enough, reconfiguration of network resources is initiated. The time between runs is based on the time required running the optimization model, which usually is slow.

This study case of self-optimization considers Carrier Ethernet networks using MSTP (Multiple Spanning Tree Protocol) based technologies. In this context the goal of the optimization model can be either minimizing the number of allocated spanning trees or the amount of allocated capacity.

6.1 Optimization of network resource allocation

A centralized entity in the network knows the actual network state, i.e., the topology, the carried traffic and the network resource allocation. For each new traffic demand arriving to the network, a new resource allocation is necessary. Optimal allocation could be achieved through complex algorithms that use all the “historical” information about traffic and topology, but they are too slow to be applied upon the arrival of each new traffic demand. Instead other simple algorithms (“online”) can be used, which are faster at the cost of being not optimal because they are based on “snapshot” information.

Online routing results in a resource allocation that is not optimal (in the case of MSTP based technologies, it means either the number of allocated spanning trees or the amount of allocated capacity are more than necessary). An optimal allocation (the minimum) could be achieved by running periodically the optimization algorithm and then deciding whether to do reconfiguration.

The use of optimization models for network resource allocation is optional. If it is not used, the resource allocation provided by online routing becomes not optimal and the network is run inefficiently. If used, a human administrator decides when to run the optimization model (which usually is slow), and whether to perform reconfiguration if the actual resource allocation is too far from the optimal. Human intervention implies higher operational costs.

6.2 The proposed scheme for self-optimization

Self-optimization of network resource allocation in MSTP provides two basic benefits, the minimization of network resource allocation (either the number of allocated spanning trees or the amount of allocated capacity) and the reduction of operational costs. The optimal resource allocation is obtained automatically, without human intervention, through the following way:
Topological and traffic information is constantly communicated from network nodes to the centralized entity.

Periodically (the time period is chosen by the operator) the optimization model is run for either minimizing the number of allocated spanning trees or the amount of allocated capacity (this is chosen by the operator). Then the calculated optimal resource allocation is compared with the actual one, and if their difference is large enough (the threshold is chosen by the operator), reconfiguration is started.

If reconfiguration is started, the proposed reconfiguration is communicated from the centralized entity to the network nodes.

Therefore the proposed solution requires the following:

- It requires an infrastructure for the communication between the centralized entity and the network nodes.
- It requires the operator to define how often the optimization model is run, which threshold would trigger the reconfiguration, and whether to optimize either the number of allocated spanning trees or the amount of allocated capacity.
- It requires an optimization model for network resource allocation in networks based on MSTP technologies. Specifically, an algorithm for optimizing either the number of allocated spanning trees or the amount of allocated capacity. The information that this algorithm would need is the network topology and the actual traffic matrix, provided by the ingress nodes to the centralized entity, each time there is a traffic or topology change. The output is the optimal allocation.

Next section is devoted to describe the optimization model for MSTP that we have designed.

6.3 Optimization model for MSTP based technologies

In this study case we use optimization models based on Integer Linear Programming (ILP) for routing and resource allocation. We consider Carrier Ethernet networks based on MSTP technologies, although we also include in the study, for comparison, those technologies based on label forwarding (i.e., ELS - Ethernet VLAN Label Switching-, PBB-TE - Provider Backbone Bridges Traffic Engineering -).

We start by reviewing the related work on optimization models in Carrier Ethernet. In the case of label forwarding based technologies, the work done in the design of ILP models to optimize resources (for technologies like MPLS) is substantial (e.g., [14]), and there is no need to propose new models since the existing ones can be applied without any modification. In the case of MSTP based technologies, the work done in the design of ILPs presents, to the best of our knowledge, the following two shortcomings:

- In all the studies, ILP models are always compared either among themselves or against the use of basic native Ethernet protocols.
- There is not any ILP model that guarantees a complete global optimum. The existing models rely on a heuristic to calculate part of the solution. For example, the model proposed in [15], uses a heuristic to pre calculate a set of spanning trees.

Following the previous related work, our goal is to propose an ILP model for routing and resource allocation in MSTP based technologies, which obtains the set of spanning trees that accommodate traffic to network resources achieving a guaranteed optimal allocation.
The tackled problem can be stated in the following way. Given a maximum number of trees $\text{max}t$, a network graph $G = (N; E)$ and a traffic matrix $\text{TM} = N \times N$, where $N$ is the set of nodes, and $E$ the set of links, the goal is to find a set of undirected trees $T (|T| \leq \text{max}t)$ and accommodate the traffic described by $\text{TM}$, so that the traffic is routed through the paths given by $T$, and the accommodated traffic is maximized.

The ILP is based on the multi-commodity flow problem. For each pair of nodes $(s, d)$, where $\text{TM}(s,d) > 0$, we refer to a commodity $c \in C$ such that the requested bandwidth of the commodity $\text{BW}(c) = \text{TM}(s,d)$ and the destination and source of $c$ are $s, d$, respectively.

Based on this, the proposed ILP consists of the following indices:

- $i, j$ for representing nodes in the network
- $c$ a commodity given by $\text{TM}$

And the following parameters:

- $\text{BW}_c$ set of requested bandwidths given by $\text{TM}$
- $S_{(c,i)}$ is set to 1 if node $i$ is the source of commodity $c$, -1 if it is the destination and 0 otherwise.
- $C_{(i,j)}$ capacity of a link
- $\text{max}t$ maximum number of spanning trees

The variables used in the model are the following:

- $f_{(i,j)}^{c,t}$ represents the amount of bandwidth accommodated for commodity $c$ on link $(i,j)$ as part of the tree $t$
- $x_{(i,j)}^{t}$ is 1 if link $(i,j)$ belongs to tree $t$, 0 otherwise
- $y_{i}^{t}$ is 1 if node $i$ is the root of tree $t$, 0 otherwise
- $h_{i}^{t}$ represents the height of node $i$ in tree $t$

In order to ensure that each tree $t$ has no cycles and is connected, the trees are modeled as unidirectional hierarchical trees, each tree has a root, and the root has height 0. If link $(i,j)$ belongs to tree $t$, then $h_{i}^{t} - h_{j}^{t} = 1$, this property ensures that there are no cycles in the tree. Regardless of the fact that the trees are modeled unidirectional, the flow constraints are designed to consider them bidirectional.

The objective function is to accommodate as much bandwidth as possible through the entire network.

\[
\text{MAXIMIZE} \quad \sum_{j \neq d} f_{(j,d)}^{c,t} \quad \forall i \mid S_{(c,i)} = 1
\]

\[
\text{SUBJECT TO}
\]

Routing constraints:

\[
\sum_{c \neq f} f_{(j,i)}^{c,t} \leq C_{(i,j)} \quad \forall i, j
\]

\[
\sum_{j \neq d} f_{(i,j)}^{c,t} \leq \text{BW}(c) \quad \forall i, c \mid S_{(c,j)} = -1
\]

\[
\sum_{j \neq d} f_{(i,j)}^{c,t} \leq \text{BW}(c) \quad \forall i, c \mid S_{(c,i)} = 1
\]
Constraint 2 ensures that the accommodated traffic on a link does not exceed the link capacity. Constraints 3 and 4 ensure that the accommodated traffic does not exceed the demanded traffic. Constraints 5, 6 and 7 are the flow conservation constraints.

Tree shape constraints:

\[
\sum_j f^{(t)}_{(j,i)} = 0 \quad \forall i, c \mid S_{(c,i)} = 1
\]  \hspace{1cm} (5)

\[
\sum_j f^{(t)}_{(i,j)} = 0 \quad \forall i, c \mid S_{(c,i)} = -1
\]  \hspace{1cm} (6)

\[
\sum_j f^{(t)}_{(i,j)} - \sum_j f^{(t)}_{(j,i)} = 0 \quad \forall i, c \mid S_{(c,i)} = 0
\]  \hspace{1cm} (7)

Constraint 8 ensures that nodes with height zero are only nodes that have no father in the tree, which are either the root or a node not belonging to the tree. Constraints 9 and 10 ensure that the difference between the height of two connected nodes in the tree is 1. Constraint 11 ensures unidirectionality. Constraints 12 and 13 ensure that there is only one root per tree and that the root is connected to at least one node. Constraint 14 ensures that the root does not have a father, and the other nodes do not have more than one. Constraint 15 ensures that a node that is not the root and does not have a father is not connected with any node.

Tree-flow constraint:

\[
f^{(t)}_{(j,i)} \leq MAX_{c}(BW(c))(x^{(t)}_{(i,j)} + x^{(t)}_{(j,i)}) \quad \forall i, j, c, t
\]  \hspace{1cm} (16)

This constraint ensures that the accommodated traffic follows the paths given by the trees. Note that the expression \((x^{(t)}_{(i,j)} + x^{(t)}_{(j,i)})\) ensures that even though the trees are modeled unidirectional, traffic can flow in any direction given by the links belonging to the tree.

More details of the proposed ILP model can be found in [16].
6.4 Evaluation results

In this section we evaluate the performance of the proposed optimization model for MSTP based technologies in terms of the total accommodated traffic. We use different network topologies and traffic patterns. We also compare these results with the ones obtained with those network technologies based on label-based forwarding.

Two types of topologies have been used, grid topologies (for example in [17]) and defined topologies (for example in [18]). Specifically, two topologies are considered in this section: a grid topology of 36 (6 x 6) nodes and the defined cost266 topology [19]. For both topologies link capacity is set to 10 Gbps. For all the experiments it is assumed that all nodes are sources and destinations, meaning traffic is generated among all nodes.

This is an offline scenario in the sense that all traffic is known in advance and used in the optimization model presented in the previous section. The traffic between source and destination is uniformly distributed between [100, 1024] Mbps. For modeling the label based forwarding technologies the proposed models is modified in the following way: the variable \( c_{i,j} \) is replaced by \( c_{i,j}^{(k)} \), and the rest of the variables are removed. Additionally all the tree shape and tree-flow constraints are removed. In order to perform a fair comparison, the objective functions are the same but using the replaced variables. The models are solved using the Xpress-Optimizer [19].

Performance is evaluated in terms of the accommodated traffic. The accommodated traffic is the sum of the amount of traffic that is routed through all the sources and destinations, and it is the objective function of the proposed models. This value is measured against the number of allowed trees \( \text{maxt} \) parameter specified in the model. The results for label-based forwarding approaches, given that they are not subject to the maximum number of trees \( \text{maxt} \), are plotted as a constant horizontal line among the number of trees. This means that only one value is calculated for the label-based forwarding per plot. On the other hand, for the STP-based, one value (represented as a point in the line) for each of the different number of trees \( \text{maxt} \), is calculated per plot. The results are presented in Figure 6.1.

Results show that when using just one tree the optimal performance of the STP-based approaches is between 36% (grid) and 41% (cost266) less than the label-based forwarding ones. The minimum number of trees that give the same performance than label-based approaches is 70 and 110 for the cost266 and grid topologies, respectively. If the total accommodated traffic is divided by the number of trees, then the average accommodated traffic per tree is 4381 Mbps (for grid) and 5406 Mbps (for cost266). This means that even thought in the grid topology more traffic can be routed, in the cost266 topology more traffic can be routed per tree.

Summing up, our proposed model successfully solved the stated problem for two topologies, and the results showed that the model can be used to determine the minimum number of VLANs the network must support in order to route a specific traffic matrix. Results also show that an optimal use of multiple spanning trees can make the MSTP based approaches accommodating the same amount of traffic than the label-based forwarding ones.
Figure 6.1 - Traffic accommodated for offline scenario.
7 Inter-domain Traffic Engineering for balanced network load

7.1 Network model

7.1.1 Reference network model

Our proposal is based on the presumption that both the aggregation and access domains are, or in the near future are expected to be, based on switched layer 2 (L2) technologies [12], which offer lower bit costs [13]. L2 switched networks deploy Spanning Tree Protocol variants (STP, MSTP, etc.) [14] to convey the traffic towards the core.

Figure 7.1 gives a picture of the reference network model developed within the CELTIC TIGER2 project [15]. The reference network [16] reflects the view of major service providers and vendors on the evolution of networking infrastructure and the way it will assimilate the new technologies. As seen in Figure 7.1, the networks are divided in three segments: the Access, the Metro and the Core. Depending of the country and/or local geographic specificities as well as the Internet Service Provider (ISP) choices, part of the sub-segments depicted in Figure 7.1 may be missing, but based on the current practices and medium-term forecasts, this generic model describes all networks.

![Figure 7.1 - TIGER2 generic network reference model.](image)

The Access network is a local area network, and is widely studied in the literature. It connects the end users to the first Central Office (CO). Typically they have a tree-based topology, which aggregates the traffic to the COs. Core networks are also well studied and in this model we define it as the national or wider area domain. Typically they have a meshed topology. As seen in Figure 7.1, the metro network, which links the access to the core, is split into three sub-segments. In legacy infrastructures, these sub-segments together form a hierarchy. Access areas may be connected to any metro sub-segment by COs, and each metro is connected to the core through a Point-of-Presence (PoP).

The above model is way too generic for analysis though. The roles of the sub-segments should be specified in the context of the deployed technologies. In this paper we assume that carrier-grade Ethernet-based layer 2 technologies become dominant not only in the aggregation, but also in the access [12]. Based on these assumptions we obtain the particular reference network presented in Figure 7.2, derived from the generic network model [16].
In this specific model the metro sub-segments use layer 2 (L2) switching in the access and edge, while the core deploys L2/L3 TE mechanisms. Thus, the first two sub-segments of the metro represent two successive aggregation levels of the user traffic. In the metro-access, the first aggregation level, the traffic from multiple COs (Central Office) is aggregated in Concentration Nodes (CN). This segment is based on a dual star topology with GbE or 10GbE links between COs. As seen in Figure 7.2, each CN aggregates up to 15 000 nodes. In the metro-edge, the second level of aggregation, traffic from different CNs is processed by a L3/L2 metro node, and the PoPs at L2/L3 boundary are handling up to 50 000 users. As a summary, we can say that metro-access and metro-edge are pure aggregation networks, while the metro core segment presents a meshed traffic distribution.

### 7.1.2 Core network models

A specific core network model has been proposed [16], starting from current ring topologies, widely deployed in optical networks. The model is a Double Rings with Dual Attachments (DRDA) and it can be used in core networks. In such topologies two rings, (the inner and the outer metropolitan rings) are interconnected in such a way, that every node in the outer ring is directly connected with its associated node in the inner ring, via double links (dual attachment). These provide high connectivity and multiple back-up paths for restoration purposes while reusing current network fiber deployments.

### 7.1.3 Investigated network topologies

Based on the reference networks presented in the previous section we designed a network that was used for our simulation based investigations, and its topology is presented in Figure 7.3 (left). This network is divided in two main parts, an aggregation network using Multiple Spanning Tree Protocol (MSTP) [14] and a core part with Constraint based Shortest Path First (CSPF) TE [10].

The traffic sources are depicted on the left-most part of the figure, the aggregation network conveys the packets to the core network. At the boundary we have only three edges. In real-life networks the number of edges is kept as low as possible for reasons of costs. The network has six destination nodes (sinks) represented by the exit points of the core network on the right side. The main function of the aggregation domain is to channel end-user traffic towards the core, thus its nodes are connected to two neighboring devices, at most.

The core domain has a meshed topology, with a 3 hop shortest distance between the ingress and the egress. The nodes of the core have a degree of connectivity of 3 or 4. This is a trade-off between cost effectiveness and the assurance of alternative paths. The aggregation domain
uses Ethernet-switched technology, and the core uses WDM extended with an electronic control layer.

Apart from investigating the efficient network capacity usage and balanced load of the core domain, we also investigated the possibility to minimize the operations in the electronic layer and the usage of longer optical paths. These last two parameters are characteristics of the dual opto-electronic models.

![Figure 7.3 - Topologies of aggregation and meshed core](image)

Our proposal supposes that the domains have a control plane that apart of running TE and other control functions are capable of communicating/cooperating with the control planes of the neighbouring domains. Such a control plane model is the Knowledge Plane [23] that can use MSTP in the aggregation and CSPF in the core domains.

We will also investigate the behavior of the core if it deploys a dual ring topology. We will build that topology in such a way as the edge nodes and the output nodes from the previous topology will be kept in order to use the same aggregation network and to be able to compare the two results.

### 7.2 The proposed solution: Inter-domain TE cooperation

Our proposal is to use shared intelligence between control planes, where the core intra-network functions are unchanged and only the inter-network control planes co-operate which enhances the performance.

In Figure 7.3 the traffic reaches the core network through the aggregation domain. In case of any event (congestion on a link, link failure, etc.) the classical TE works with the assumption that the traffic matrix remains unchanged and it has to re-distribute the traffic volume relying on load redistribution inside the core. Our proposal is to use the Knowledge Plane and re-arrange the input traffic distribution outside the core edge routers. This means that –from the point of view of the core– we change the traffic matrix, since the load on the edges will be different.

Let us take the topology presented in Figure 7.3. Now in the situation when the aggregation domain directs all the traffic to the e1_edge (the “northern” one), while e2_edge (the “middle” edge) and e3_edge (the “southern” edge) do not feed any traffic to the core. This is the worst case situation to overload the core and corresponds to the situation when only the tree rooted in e1_edge is used to collect the traffic in the aggregation domain. Now, if we take the opposite situation, when we use each of the trees in the aggregation domain to forward the same amount of traffic, then the aggregation domain distributes the traffic evenly among the three ingresses. In this case all regions of the core will be evenly loaded.
It is the task of the Knowledge Plane to map the traffic sources among the trees. In our simulations we used small individual flow throughputs. Each tree is collecting such individual demands and the sum of these represents the traffic load at the edges. Practically the granularity of the traffic is small enough to allow us to finely balance the load. In what follows we will use the term load balancing as the operation of load redistribution in the aggregation domain as described above. The goal of load balancing will be to decongest a certain area of the core network with a minimal redistribution of the original load.

7.3 Proposed simulation model

7.3.1 Traffic model

During the simulations the traffic flows originated from the sources have the same bandwidth. We consider that we know the traffic matrix and the paths in the core are computed by a PCE using CSPF protocol. Additionally we will generate background traffic, as well, which enter the core at the edge nodes and sink on the most right-hand side destination nodes. The links of the core networks will have 200 Mbps capacity, which defines the load region where the core network is congested, but not overloaded of 400 Mbps to 800 Mbps.

In our investigations we will use the e2_edge node where we directed all the traffic and tried to serve it using CSPF. The resulting paths are called the main branch. If the demand is high enough, the traffic demand cannot be served. If we will apply our solution to this situation that means that some part of the traffic will be shifted to the other two edges, e1_edge and e3_edge. The paths that follow the flows entering on these two edges are called secondary branches.

We will use the background traffic to “fill” the network up to the point where congestion might start to develop. We will send 200 Mbps background traffic on the main branch. Then we will start to add new traffic demands until we will reach the total one, which will be set differently from case to case: all our simulations are planned to be run with the 500Mbps, 600Mbps, 700Mbps and 800Mbps total traffic demand. These are the situations when we can test the usefulness of our proposal and evaluate its impact on the efficiency of the opto-electronic core transport.

We use a flow level simulator, already used for the research of opto-electronic networks [24]. We generate individual flows, and the sum of these demands result in the overall traffic demand. Each link is divided into lightpaths of 10 Mbps capacity. This results in 20 lightpaths within each link that offers enough flexibility for multiplexing the flows within the core. Based on earlier work with the simulator we opted for 12 individual flows per lightpath, resulting in a flow capacity of 0.83 Mbps.

Within each scenario—that is for different overall traffic demands— we have simulated several sub-cases, where the load of the main branch was gradually re-distributed among the secondary branches. At first we started with the situation where 30% of the traffic was entering at edges e1_edge and e3_edge (15% on each of them). From there on, we stepwise directed more and more traffic towards to the secondary branches while the network was able to carry the traffic without loss. In order to be sure on that, we also simulated the next step following this point. The individual flow demands were scheduled randomly. For each situation we run ten simulations and averaged the results.
7.3.2 Spanning Trees in the aggregation domain

In order to get the input traffic at the ingresses of the core domain, we had to build the trees that bring the traffic to the edges of the core. For this, we had to build the set of trees that form the basis of the MSTP operation. We used a combination of the TOTEM [25] and BridgeSim [26] tools to simulate these trees.

With the combination of these two tools we could use the topologies created in TOTEM and apply the STP formation protocol implemented in BridgeSim. We generated all the spanning trees that can potentially be used in our scenarios, that is, all the trees that are rooted in one of the three edges and the traffic sources are their leaves. Figure 7.4 presents the tree generated for the northern edge, the other two trees have a similar shape. The actual traffic distribution among the three trees is decided according to our solution and should be enforced by the Knowledge Plane, as mentioned earlier.

We will not investigate the behavior (delay, blocking, packet loss, etc.) of the aggregation domain, only used it to generate the MSTPs and determine the input traffic for the core domain. In our investigations we will be interested only about the traffic distribution among the three ingress nodes.

7.4 Planned work

7.4.1 Dual ring core topology

We will not investigate the behavior (delay, blocking, packet loss, etc.) of the aggregation domain, only used it to generate the MSTPs and determine the input traffic for the core domain. In our investigations we will be interested only about the traffic distribution among the three ingress nodes.
Within TIGER2 a specific core network model has been proposed [71], starting from current ring topologies, widely deployed in optical networks, the Double Rings with Dual Attachments (DRDA). In such topologies two rings, (the inner and the outer metropolitan rings) are interconnected in such a way, that every node in the outer ring is directly connected with its associated node in the inner ring, via double links (dual attachment). These provide high connectivity and multiple back-up paths for restoration purposes while reusing current network fiber deployments.

As mentioned in Section 7.1, we have prepared a network that has a DRDA topology (see Figure 7.5). We have kept the edge nodes and the output nodes from the mesh topology depicted in Section 7.1.3, in order to use the same aggregation network and to be able to compare the two results. We will refer to this network as dual ring core.

### 7.4.2 Load balancing as a result of inter-domain communication

We have conducted extensive simulations, investigating the behavior of both the mesh and dual ring core topologies, while the traffic arriving from the aggregation network was gradually changed. The traffic model has already been described in Section 7.3.1 and the logical structure of the aggregation domain in Sections 7.1 and 7.3.

![Figure 7.6 - The successfully served flow demands as the function of traffic redistribution for the meshed core (left) and the dual ring core (right).](image)

The left-hand side of Figure 7.6 presents the ratio of successfully served traffic demands in the meshed core. It can be seen that 500 Mbps total traffic will be served if we redirect 35% of the traffic on the secondary branches. The traffic volumes that must be redirected to achieve a loss-free ratio for the 600Mbps, 700Mbps and 800Mbps traffic scenarios are 50%, 60% and 70%, respectively. These results confirm that if we redistribute the traffic before it hits the core edges, we can balance the core load, thus it is a viable mechanism to actively increase the efficiency of the core traffic engineering process.

We achieved similar results for the dual ring topology, as well (right-hand side of Figure 7.6). The congestion-free core is achieved for the redistribution of the 35% of traffic for the 500Mbps case and 70% for the 800Mbps (worst) case.

Based on the above results we can see that if congestion occurs in the core, we can eliminate it just with a proper coordination between the control planes of the aggregation and core domains, redistributing the traffic prior entering the core.

In our simulation model we used MSTP protocol in the aggregation domain and the traffic redistribution was done using these spanning trees. This solution increases the ratio of
successfully served traffic demands, increasing the utilization of the core. As a conclusion we can say that the cooperation of the control layer of the aggregation and core domains has multiple advantages.

### 7.5 Conclusion

In this chapter, we have presented in detail the motivation and the envisaged environment in which our proposal on inter-domain cooperation for improved TE of the core networks would help the operators to increase the efficiency of their networks. We presented our solution and we prepared a simulation model, defined the network topology based on the TIGER2 reference network. We defined the traffic model, and after a set of preliminary simulations we identified those traffic conditions, which simulated in detail will help in evaluating the proposed solution. Finally, we conducted these simulations and demonstrated the advantages of inter-domain cooperation in increasing the efficiency of operators' networks.
8  Traffic Analysis for the Knowledge Plane

Traffic analysis of network segments is an effective method to reveal suboptimal configuration, hidden faults and security threats. If the analysis results are promptly acted upon, improvements in service quality are experienced by both the network operator and the end-user. The concept of the Knowledge Plane (KPlane), and later the Monitor Plane (MPlane) has been introduced to support Autonomous Networking goals. The tasks of processing the network element-, service-, and traffic-information belong to the MPlane. It feeds the KPlane with valuable information, based on which configuration changes are actuated. Although the concept of KPlane is widely used in various levels of network and service management, general traffic analysis is not yet utilized to support decision making procedures. Traffic mix and traffic matrix analysis results are of major interest in the decision making process at the KPlane.

8.1 Problem Statement

The optimization of network and service resources and the maximization of end-user experience are not necessarily conflicting terms. The reason for such belief lies in the fact that current network operators and service providers lack of up-to-date, usable information on their traffic. The questions of “how much” of “what” actually are traversed on the various network segments, where is that traffic “originated from” and where is it “distributed toward” are rarely answered.

According to the main argument of [21], the users and the operators suffer from the lack of a serious, purposeful optimization effort in the traditional Internet. The transparent core has no knowledge about the data transported, and even if the intelligent edge nodes realize that there is a problem, the core might not be aware of what should be done. The low-level decisions (at the edge) are rarely relate to the higher-level goal (of the core). On the user side this results in meeting the service level agreement only in coarse granularity: it is measured in long periods and more at a network level, rather than on a per-service basis.

The solution for gaining knowledge about network status and traffic characteristics is to gather and process such data, which then provide a basis to trigger corrective actions. The authors of [21] suggest to handle this knowledge in the Knowledge Plane (KP), an abstract entity that completes a triad together with Data Plane and Control Plane (see Figure 8.1).

In the original KPlane concept, the input is taken by sensors and the output is given by actuators. A practical variation of this architecture, detailed in [22], splits the KPlane into monitoring plane and knowledge plane. The separation of those is an obvious step: the actual “network monitoring units” (sensors) that capture and pre-process traffic data represent the “monitoring plane”, similarly as depicted by Figure 8.1. There are further variations and additions to this architecture; we will review these in the section of Related Works, together with a short review of decision making methodologies and practical examples from the field.
Figure 8.1 – Functions of the Knowledge Plane and its connections to Control and Data Planes

Figure 8.1 depicts the relation between the Knowledge, Control, and Data Planes. The probes/sensors take data from both the control and data planes, and report pre-processed information for the status processing module, where further analysis takes place. The actuator in the model is the decision maker module, which provides triggers for the control plane, completing the self-management cycle.

The main source of “knowledge” is the actual traffic of the Control and Data Planes. Although some traffic characteristics can be gathered by analyzing the Control Plane messages, many important applications – such as Peer-to-Peer (P2P) downloads, Video Streaming, or interactive voice – hide their control messages, hence their identification is only possible through Deep Packet Inspection (DPI) of the traversed traffic. The aim of Traffic Mix analysis is to determine the distribution of volumes for services and applications utilizing the network. Similarly, Traffic Matrix analysis provides results about traffic volumes – and if possible, further characteristics – broken down by route directions.

It is clear that the concept of Knowledge Plane is widely used in various levels of network and service management. Nevertheless, general traffic analysis is not yet utilized in order to support decision making in the KPlane. In the following sections we describe the suggested management architecture, traffic analysis concept and two methods to extract valuable information about the traffic mix and the traffic matrix.

8.2 Solution

8.2.1 The Monitor Plane

We follow the architecture suggested in [23] (see Figure 8.2), and closely examine the functions and requirements of the Monitor Plane. This function is crystallized at the original definition of autonomous networks, in [25], defining the foursome of “Monitor-Analyze-Plan-Execute” (MAPE) functions. The core function of the MPlane is to provide complete and detailed view of the network and its services. Probes at every element (access nodes, routers, switches, content servers, links, etc.) monitor the element status as well as traffic parameters.
Although built-in probe modules seem convenient, passive probing is more desirable. Active network elements (such as routers or switches) keep their processing priorities to their main job, occasionally leaving the Knowledge Base without information. These occasions of degradation in the status reporting function happen at the worst time from the KPlane’s point-of-view – for practical reasons. It gets degraded at the time when the element is getting overloaded. Coincidentally, such detailed reports of overloading would be the most beneficial for the KPlane. This is why passive probing is more desirable to gather information on these elements.

After capturing the raw data, processed, grouped, and filtered traffic information gets inserted into the Knowledge Base by the probes. Both packet- and flow-level analysis reveal important characteristics on losses, delays, and jitters in the traffic, routing specialties, network structure changes and violations of the SLS (Service-Level Specification).

We are focusing on gathering these characteristics by passive monitoring. In the following subsections we briefly describe the basic requirements and mechanisms enabling this method.

### 8.2.2 Basic Functions of the Probes

The inevitable function of the network monitoring probes is catching, filtering, and preprocessing the traffic. These tasks should be completed for the whole network. Since installing and maintaining such a monitoring network could be an enormous effort for the operator, introducing the MPlane at the highest aggregation parts (i.e. monitoring the fastest links) can be a good decision. Monitoring these relatively few points allows gathering all packets that traverse the network, although some locally looping traffic could be left out of the analysis.

The probes should have the following crucial functionalities:

- Creating timestamps for the packets. Time-stamping done by hardware (firmware) facilitates much more precision than by software, since it avoids possible latencies due to the operating system.
• Filtering on hardware level. High-speed traffic (i.e. currently 10 Gbps or above) presently allows no option for on-the-fly filtering in software. Clearly defined, low level filters are very useful: they can dramatically decrease the data to be analyzed.

• Truncating incoming packets. For the majority of the network analysis functions, statistics-counting, or fingerprint analysis, it is not necessary to use the whole IP packet, but the first portion of it. A practical example is truncating at 128 bytes, which keeps TCP and IP headers as well as the beginning part of application headers that are helpful for identification, since it contains fingerprints for P2P or video.

• Traffic processing. The main traffic processing functionalities are briefed in the next section.

• Encapsulation and presentation of preprocessed data. The traffic analysis results must be structured and packed when passed over to the Knowledge Base.

8.2.3 Traffic Processing

The time-stamped, filtered, truncated packets, must be processed in order to reveal network and service statuses. Depending on the traffic volume, and the depth of the analysis, this processing can be fed into one or many processors. In order to keep up with the ever increasing traffic and the demand for complex analysis, the processing system must be highly scalable. As discussed earlier, monitoring core links has the advantage of utilizing all through-traffic (that traverses the network), although it requires equipment being able to monitor high-speed (currently 10 Gbps Ethernet) links without frame loss.

Figure 8.3 – A 10 GEthernet-capable, lossless network monitoring card, SGA10GED

For low analysis demand (when one CPU can deal with the challenges), a highly reliable monitoring card, such as SGA10GED can be used to capture the traffic. It fits into a PCI slot of an industrial grade PC, where it captures, timestamps, filters, and truncates packets before passing it to the main CPU where Traffic Analysis is performed. Figure 8.3 shows an SGA10GED card.

In cases where on-the-fly, complex analysis is required on highly utilized links, the SCALOPES C-board is a highly scalable solution. It is a standalone, FPGA-based hardware,
equipped with 2x 10 Gbps Ethernet interfaces and 16x 1 Gbps Ethernet interfaces. When used as part of the Monitor plane, it is also preprocessing the packets, but rather than passing their data to one CPU, it distributes them among many monitor units through its 1 Gbps Ethernet Interface. The standalone Monitor Units then carry out traffic analysis, and present the results to the knowledge base. Figure 8.4 depicts such a scenario. Detailed description of this system can be found in [26].

The distinct analysis tasks – such as flow separation, application identification, QoS-related parameter calculation per flow/application/route – are managed by separated modules, so the parallel tasks can be run on distinct processors in the same time. Moreover, the inactive modules can be turned off to save power.

Figure 8.4 – A scalable solution for Traffic Analysis of high-speed network links

The tasks of the monitor units in this architecture are the following:
- collect and decode all the incoming information continuously (in 7/24 manner)
- check filtering rules predefined by the network operator, execute conditional controlled orders/commands (conditional packet saving, alarming)
- structured data storage (raw data, statistics, assays, alerts)
- generation of packet- and flow-level counters on volume, loss, delay, jitter
- generation of specialized traffic reports, such as traffic mix and traffic matrix
- database handling, remote access/query (Remote Capture, Session/Flow Trace)

8.2.4 Decision Making

Since processing of network status is continuous at the KPlane, and faults/attacks may happen at any time, so decisions on corrective actions have to be made on-the-fly as well. The Action Plane should be notified (instructed) about these actions for execution. Although the accuracy of decision making process is the key, it is limited by the variety of the input information – which is in this case merely traffic-related. Beside the accuracy, speed is also a key factor.

In order to understand the complexity of the decision making problem, a short review the main challenges are necessary. Clark et al. [21] points out three significant issues that needs to be addressed by the Knowledge Plane.
- The KPlane needs to operate in the presence of incomplete and inconsistent information, with the possibility of even misleading or malicious pieces of data.
- The KPlane needs to be able to handle conflicting or inconsistent high level goals.
• The KPlane needs to be general and future proof, i.e., the introduction of new technologies and novel applications should be possible. Moreover, the environment in which optimization needs to take place is highly dynamic, where both short and long term changes are possible in the structure and complexity of the network system.

Such challenges are not uncommon in the research and applications of the last decades of Artificial Intelligence (AI) literature. In particular, multi-agent systems (see [33]) are often proposed to handle such challenges. A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents, where intelligent agents, shortly put, mean autonomous decision making entities with individual information processing capabilities and individual goals. Such agents can naturally incorporate different viewpoints or goals in a system and also provide a natural way to embody components with different levels of data access. As a consequence, however, the goals and actions of agents in a multi-agent system may partially be aligned or conflicting. Also, even if conflicts are missing or resolvable, information may be unevenly distributed among the agents. Therefore, agents interact and try to resolve conflicts and collaborate according to various protocols and methods. A vast body of the recent AI and MAS literature deals with conflict management, collaboration and cooperation, and distributed optimization in such systems (see [34], [35], and [36]).

It is worth pointing out that the agent metaphor is a natural abstraction layer to describe conflicting or inconsistent goals – independent of the particular problem at hand. This is also true for matters of trust (c.f., malicious information). This way, these issues can be handled by general solution methods and need not be developed for each particular application domain. In other words, these challenges of the Knowledge Plane may be handled by “canned solutions” developed in other research domains.

Multi-agent systems are often said to provide a solution for the introduction of novel applications as well. The idea behind this proposal is that if a new application or requirement appears, a new agent (or bunch of new agents) may be introduced to the system at any point in time. With the general conflict resolution and collaboration protocols in place, the new goals and requirements represented by the new agents will be seamlessly integrated in the system. Similarly, should some of the goals rendered outdated by time, the sets of agents can be gracefully eliminated from the multi-agent system.

Still, in order to proceed towards a decision making solution in the autonomous networking field, further research is required. Although recent AI-related research should be exploited in the area of network management, currently there are no real-time, scalable solutions available. The canned multi-agent solutions have not yet break into the network management field, and the few prototype systems (e.g. the one described in [37]) remained prototypes up to now. Due to the above mentioned limiting factors, a scalable, high-performing, yet less accurate solution is suggested for decision making: rule-based reasoning. It is used with success in many areas; see [24] as an example. In connection with the KPlane, we continue future research in the AI-field, and further developments and integration toward a scalable, rule-based reasoning engine that is applied in a distributed manner throughout the KPlane.
8.3 Traffic analysis

8.3.1 Traffic Matrix Calculations

Traffic Matrix is a network planning and development tool. During Traffic Matrix analysis, basic QoS statistics are periodically created on flow-level, and matched to originating and destination routes, network segments, or endpoint pairs (such as IP address(-range) pairs, MPLS tunnel endpoints, etc.). The first step of the analysis is determining the flows by an n-tuple (i.e., “5-tuple”: from-IP, to-IP, from-port, to-port, protocol), and building/refreshing the flow-database. Once the targeted data structure is clarified, the algorithms of Traffic Matrix calculation are of low complexity. Such algorithms are described in [27]. The result of the measurement can be used to display periodical statistics that support network planning or service marketing activities.

The actual Traffic Matrix can easily contain endpoint-pairs in the magnitude of $10^5$. It is challenging to display such huge amount of data in a way that humans understand. While the raw results should be made available for reference in the Knowledge Base, some kind of data grouping should also be applied for visual presentation. One example of a good solution is to group the matrix elements into network segments, based on their destination addresses. The aim of the grouping algorithm is never to display high, invisible amount of segments (e.g. more than 15). When the operator wishes to peek inside a segment’s statistics, he/she get it displayed as a deeper layer of the matrix. This way the calculated QoS parameters show up in an aggregated manner in the segment-to-segment relation. If the system allows manual definition of segment-creation rules, operators can gather valuable information by grouping their endpoints into various segments. An example screenshot from a solution integrated in our system is shown by Figure 8.5.

![Traffic Matrix visualization application](image)

**Figure 8.5 - Screenshot of a Traffic Matrix visualization application ([28])**
8.3.2 Traffic Mix Statistics

Traffic mix analysis is the classification of traffic flows into application types, and then evaluating these for the service parameters important for the given application type. Flows are classified by means of statistical indicators and, if necessary, behavior heuristics. The most important flow types include video stream, video conference, or simple download of videos, audio stream, VoIP, and peer-2-peer.

An application belonging to a traffic-class can be identified by using static identifiers (e.g. port-based), dynamic identifiers (e.g. changing ports, fingerprints) or by applying packet-level statistics-based evaluation methods (i.e., Naïve Bayes). Powerful identification methods for VoIP, video and p2p applications are described in [29], [30], and [31] respectively. We used these methods successfully during the WP2 – see [32] for details.

Once a traffic flow is identified (i.e., based on 5-tuple), various metrics are calculated in order to help identifying the traffic-class. These metrics are the following:

- throughput: transferred data bytes per second,
- packet loss: the rate of received packets and total transmitted packets in a given time interval, or during the connection,
- packet delay: depending on the network topology and link load it takes a certain amount of time to receive a packet after it was sent; there is also a gap (delay) between packets on the wire,
- jitter: network load is not always static: as conditions and usage changes over time, packet delay changes as well - this is called jitter,
- round-trip-time: interactive applications require fast replies, which can be characterized with this parameter,
- out of order/duplicated packets.

Figure 8.6 – VoIP Portion visualization of a Traffic Mix analysis
Figure 8.6 depicts a partial result of one of our measurement at a major ISP. It visualizes the number of parallel VoIP sessions (upper diagram) and the traffic volume (in kbps). The different kind of VoIP traffic are represented with different colors, which are - from bottom to top - a) Skype over UDP, end-to-end; b) Skype over UDP, end-to-office; c) other type of VoIP, d) Skype over TCP.
9 Conclusion

This deliverable presented the solutions to the study cases that have been identified in the scope of TIGER2 WP4 activities, as well as their evaluation results. Figure 9.1 recalls the list of these study cases and highlights the characteristics of the proposed solutions in terms of self-* features. As mentioned in the introduction, and highlighted by figure 1.1, this document is an updated version of the D41 deliverable including detailed evaluations of the proposed study cases. The following summarizes the results obtained through these study cases:

- **Hitless maintenance**: This study case addressed the design, feasibility and performance evaluation of an automated platform aiming to orchestrate maintenance operations based on operator's objectives. A distributed algorithm and relevant metrics have been proposed and evaluated using simulations as well as the designed emulation platform (proof-of-concept prototype).

- **Greening TIGER2 architecture**: This study case addressed the design of green routing algorithms and the evaluation of the achievable energy savings that such mechanisms could allow in several realistic network scenarios. Energy-aware routing has been studied as an optimization problem, evaluating its effects on the network energy consumption and on the network device load, a standard indicator for the QoS performance.

- **LOCARN**: This study case addressed the design of a "Low Opex and Capex Architecture for Resilient Networks". LOCARN is based on two simple concepts (auto-forwarding

Figure 9.1 – Map of TIGER2 WP4 study cases

[2] The study case on "Adaptive control of Path Computation Elements" has been kept in this map. On the other hand, the study case on "Traffic analysis for the knowledge plane" has not been reported in this map as it represent a common functionality possibly required by several self-* solutions.
and enhanced broadcast) and is compliant with several autonomic networking properties. A distributed source routing algorithm for topology discovery has been then proposed and evaluated in order to improve the performance of the LOCARN architecture.

• **Self-optimization of network resource allocation in MSTP**: This study case proposed and evaluated an optimization model in order to automatically ensure an optimal resource allocation in MSTP. The goal of the optimization model can be either minimizing the number of allocated spanning trees or the amount of allocated capacity.

• **Inter-domain traffic engineering for balanced network load**: This study case proposed new traffic engineering methods, based on inter-domain cooperation and intelligence sharing between control planes, for balanced network load. The advantages of the proposed solution have been demonstrated through realistic simulations.

• **Traffic analysis for the knowledge plane**: This study case provided a relevant functionality applicable in several domains. Indeed, it addressed knowledge building through extensive traffic monitoring, traffic mix and traffic matrix analysis with the aim to support decision making. The monitor plane concept and proposed physical equipment have been then described.

Besides these promising results, the feasibility and performance of the proposed approaches, and hence self-* paradigms, in tackling network control and management issues have been assessed and demonstrated.
Annex A: Generic LOCARN Frame Formats

Figure A.1 - Generic LOCARN Data Frame structure

Figure A.2 - Generic Path Request LOCARN Frame structure

Figure A.3 - Generic Path Discover LOCARN Frame structure

Figure A.4 - Generic Hello Forward LOCARN Frame structure

Figure A.5 - Generic Hello Back LOCARN Frame structure
11 References


[34] Lander, S.E., “Issues in multiagent design systems”, IEEE Expert, April 1997


D41-2 - Design of study case solutions & study cases evaluations


[50] Ampl, a modeling language for mathematical programming, "http://www.ampl.com/".


[52] Ieee 802.3az task force, "http://www.ieee802.org/3/az/index.html"


Celtic tiger2 project website, "http://projects.celtic-initiative.org/tiger2/".

Internet2 network website, "http://www.internet2.edu/network/".


The TIGER2 D20 Deliverable, “Rationales and scenarios for investigations on next generation of access, backhauling and aggregation networks,” November 2009.