# Transfer Thesis

Daphne Tuncer
Department of Electronic & Electical Engineering
University College London
Torrington Place
London
WC1E 7JE
d.tuncer@ee.ucl.ac.uk

December 2011

# Abstract

This report presents the current state of research work that has been carried out in the context of the Ph.D. With the evolution of communication technologies and the emergence of new services and applications, the management of network resources has become a key challenge. In search for a paradigm shift, recent research efforts have been focusing on self-management principles. The Ph.D. work proposes to investigate how autonomic principles can be extended and applied to fixed networks for quality of service and performance management. In this report, a new intra-domain adaptive resource management approach for IP networks is presented. The proposed approach is so that the traffic distribution in the network is controlled in an adaptive and decentralised manner according to the network conditions. The control decisions are taken in a coordinated fashion by the set of source nodes organised into a dedicated logical infrastructure where they can interact with each others. The report discusses the main engineering and research challenges raised by the design of such an approach, and describes and explains the principles and mechanisms used to support the proposed adaptive scheme. It then presents an extensive evaluation analysis of the performance of the different mechanisms of the approach. The report then describes the future main research issues that will be investigated in the context of this Ph.D.

# Acknowledgements

I would like to thank my supervisor, Prof. George Pavlou, for his guidance and his confidence in the work that I have been carrying out in the context of this Ph.D. I would also like to thank Dr. Marinos Charalambides for his help and numerous discussions about the work presented in this report. I also thank Dr. Ning Wang for all his insightful comments and discussions about traffic engineering and multi-topology routing. I thank a lot Dr. Stuart Clayman for all his help with programming and with the different software tools. I also thank Prashant Jain for his help to perform the evaluation of the communication protocol.

# Contents

# List of Figures

# List of Tables

# Publications

D.Tuncer, M.Charalambides, G.Pavlou, N.Wang, **DACoRM: A Coordinated, Decentralized and Adaptive Network Resource Management Scheme**, to appear in the Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS'12), Maui, Hawaii, USA, April 2012.

M. Charalambides, G.Pavlou, P. Flegkas, N. Wang, D. Tuncer, **Managing the Future Internet through Intelligent In-Network Substrates**, IEEE Network Magazine, Special Issue: Managing an Autonomic Future Internet, Vol. 25(6), November 2011.

D.Tuncer, M.Charalambides, G.Pavlou, N.Wang, **Towards Decentralized and Adaptive Network Resource Management**, to appear in the Proc. of the 7th IEEE/IFIP International mini-Conference on Network and Service Management (mini-CNSM'2011), Paris, France, October 2011.

D.Tuncer, M.Charalambides, G.Pavlou, **An On-line Approach for Adaptive Resource Management in Future IP Networks**, in the Proc. of the London Communication Symposium (LCS'11), London, UK, September 2011.

Daphne Tuncer, Marinos Charalambides, and George Pavlou, **Towards dynamic and adaptive resource management for emerging networks**, in the Proc. of 4th international conference on Autonomous infrastructure, management and security (AIMS'10), PhD Workshop, Burkhard Stiller and Filip De Turck (Eds.), Springer-Verlag, Berlin, Heidelberg, 93-97.

# Chapter 1

# Introduction

## 1.1 Context and Motivation

With the evolution of communication technologies and the emergence of new services and applications, developing approaches for the management of network resources with minimum human intervention has become a key challenge.

Today networks are planned and configured in a predictive manner through off-line traffic engineering [1] where the expected demand is calculated from previous usage and a specific routing configuration is produced, aiming to balance the traffic and optimise resource usage for the next provisioning period. Reactive approaches are not possible given the current nature of management systems that are external to the network and the resulting latency in learning about arising conditions and effecting changes. As such, this external off-line configuration approach can be well sub-optimal in the face of changing or unpredicted traffic demand.

In fact, the only current dynamic intervention relates to control functionality within network devices in order to deal with faults. For example, fast reroute control functions [2] direct traffic away from a failed component upon detecting a failure. But this is done in a predefined manner that does not take into account any other aspects related to the current state of the network and can have an unforeseen negative impact elsewhere.

This lack of adaptability to arising conditions makes the current Internet unable to cope with the requirements of emerging services and time-critical applications characterised by highly demanding traffic, various traffic patterns and quality of service requirements. To meet these requirements, the future Internet is expected to become more flexible, adaptive and reactive to traffic and network dynamics. Because of the static nature of current management approaches towards these emerging demands, a paradigm shift is required and new management approaches should enable the Internet to continuously optimise the use of its resources and to recover from problems, faults and attacks without impacting the supported services and applications.

## 1.2 Problem statement

To overcome the drawbacks/limitations of current approaches, recent research efforts have been applying the autonomic computing principles developed by IBM [3] to network management systems. According to these principles, management systems are enhanced with self-aware, self-adaptive and self-optimising functionality, which is embedded within the network devices themselves. This allows the system to analyse and understand its environment, to decide upon the appropriate configuration actions, and to learn from previous decisions based on closed-loop feedback control solutions. More precisely, in order to perform closed-loop control, sophisticated monitoring mechanisms

should be in charge of gathering network information, which can be used by decision making processes distributed across network nodes. While functionalities with various degrees of automation are already implemented in today's networks and systems, these mainly apply to specific problem and to a limited extend. To support the requirements of the Future Internet, ongoing research efforts have been focussing on extending these functionalities to the various aspects of network management.

The work in this Ph.D. will investigate some of the main engineering challenges raised by the design and implementation of self-managed adaptive networked systems, through the analysis of realistic and specific in-network self-management applications.

## 1.3 Contribution

Important aspects of this work will be investigating how autonomic principles can be extended and applied to fixed networks such that continuous resource optimisation can be supported through dynamic and adaptive traffic engineering functions. Designing an approach for adaptive resource management raises several research challenges. In fact unlike current approaches, a self-managed network should support decentralised decision making and control functions. The decision making process is distributed among sets of nodes in the network, each node being responsible for deciding on the actions to take based on local feedback regarding the state of the network. The main benefits of this approach are robustness to failures and immediate adaptation to changing conditions. However, as each node has partial knowledge of the global information, inconsistencies between several independent decisions can occur, which may jeopardise the stability and the convergence of the global behaviour of the system. As such, management decisions need to be made in a collaborative manner and be harmonised toward a common objective.

The work in this PhD will address the issues raised by the coordinated decision making process between the set of nodes in the network. More specifically it will investigate how the overall system behaviour can be determined from the nodes interactions, as well as how to harmonise several decisions towards a common objective. In addition, it will investigate effective organisational and communication models to support the interaction between the nodes, as well as analyse the complexity and cost associated with those issues. These research questions will further be extended to identify generic patterns and infrastructure that can be used in different in-network self-management applications (i.e. for adaptive resource management), and to investigate how different applications realising different management tasks can gracefully co-exist within the same networking infrastructure.

## 1.4 Current work

In the context of the research efforts carried out from the beginning of the Ph.D. research work, the followings have been investigated.

A new approach for resource management in intra-domain IP networks, called DACoRM (**D**ecentralised, **A**daptive, **Co**ordinated **R**esource **M**anagement), has been developed. The proposed approach is so that the traffic distribution in the network is controlled in an adaptive and decentralised manner according to the network conditions.

The Ph.D. work has been investigated a decentralised algorithm to support the adaptive control of traffic distribution. The algorithm is executed by the network source nodes only that are equipped with a set of components to support the adaptive process. In addition, in order to guarantee the consistency between the different control decisions,

a specific coordination mechanism that takes place between the source nodes has been designed. To support this coordination process, a specific infrastructure to organise the source nodes, called *in-network overlay*, has been proposed. This *in-network overlay* is used to support the interaction between the different relevant deciding entities. Different logical organisational models of the source nodes in the *in-network overlay* have been investigated and a specific signalling communication protocol has been designed. The performance of the approach has been evaluated using real topologies and traffic datasets. The evaluation has been performed with respect to the performance of the decentralised adaptive algorithm in terms of resource usage and to the performance of the proposed approach in terms of the communication cost (i.e. message overhead and delay) incurred by the coordination mechanism.

## 1.5 Report Outline

The remainder of this report is organised as follows. Chapter 2 introduces the state-of-the art in the area of autonomic computing and autonomic network. It also presents the main current research efforts towards adaptive resource management approach, especially through online traffic engineering functionality. Chapter 3 presents the main features and mechanisms used in DACoRM. It decribes how path diversity is provided between the different pairs of nodes in the network, and introduces a logical infrastructure to support the interaction between the nodes. It also presents the main components implemented in the nodes to support the adaptation process. Chapter 4 focuses on the adaptive scheme of DACoRM, i.e. on the actual algorithm to control the distribution of traffic in the network. It also describes the principles of the coordination process between the network nodes. Chapter 5 investigates specific organisational models to support the interaction between the network nodes, as well as a relevant signalling communication protocol. It compares the different models by discussing their advantages and drawbacks. The performance of the different mechanisms used in the proposed adaptive resource management scheme are then evaluated in Chapter 6. Chapter 7 summarises the current work and presents the future research works that will be investigated in the context of this Ph.D.

# Chapter 2

# Litterature Review

## 2.1 Network self-management

### 2.1.1 Introduction

With the evolution of communication technologies and the emergence of new services and applications, the management of networking infrastructure has become a key challenge. Current management approaches mainly rely on off-line functionalities and static configurations performed through external management systems and as such, are inadequate to meet the requirements of these new applications and services in terms of quality of service, dependability, resilience and protection. To support these new services and applications, characterised by highly demanding traffic, various traffic patterns and quality of service (QoS) requirements, management approaches that can offer flexible, adaptive and reactive functionalities to traffic and network dynamics are expected.

Recent research effort have been interesting in applying the autonomic principles developed by IBM [3] to the management of network resources. The objective of these new approaches is to reduce the requirement of human intervention in the management process through the use of feedback control-loop solutions that continuously re-configure the system in order to keep it in a desirable state. More specifically these efforts focus on enabling self-management capabilities, whereby network elements can adapt themselves to contextual changes without any external intervention through adaptive and flexible functions.

### 2.1.2 Autonomic Computing

The essence of autonomic management principles have been conceptually defined by IBM in its vision of *autonomic computing* [3], that is inspired from the autonomous nervous system of the human body, which maintains an equilibrium between various biological processes in the body without any conscious effort [4]. Autonomic computing can be envisioned as computing systems able to manage themselves given high-level objectives from the network administrators, essentially towards the areas of configuration, fault, performance and security. According to this principle, four "self" properties are defined [3][5]:

- **self-configuration**: ability of the system to configure and reconfigure itself given high-level specifications, so that the system can self-organise into desirable structure and pattern.
- **self-optimisation**: ability of the system to automatically adapt to changing environment in order to optimise its performance and efficiency (e.g. its cost).

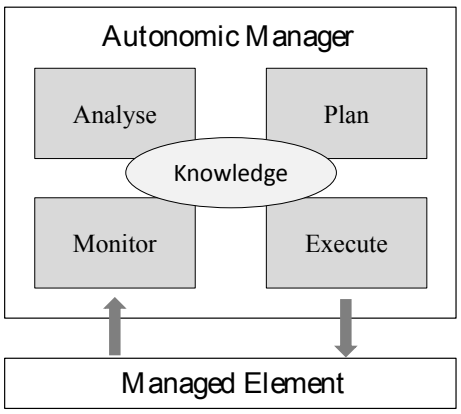


Figure 2.1: The MAPE-K control-loop

- **self-healing**: ability of the system to automatically detect, diagnose and recover from faults, either resulting from internal or external errors.
- **self-protection**: ability of the system to protect itself against potential attacks or inadvertent failures.

The *autonomic computing* paradigm is realised through the introduction of autonomic managers responsible for monitoring and controlling some managed elements and their operational environment through an autonomic management control loop [3]. Managed elements can represent any hardware or software resource such as, for instance, a storage disk, a CPU, a database, a router queue or a fault localisation service [6]. Based on the monitored information, the autonomic managers can control the current state of the managed elements and decide if any adaptation of the current element's configuration is required. More specifically, the autonomic management control loop consists of four components (**M**onitor, **A**nalyse, **P**lan and **E**xecute) relying on a common **K**nowledge repository [3] as depicted in Figure (2.1). This model is referred as the MAPE-K control loop.

The *monitor* component is responsible for collecting information from the operational environment and the managed elements. The collected information is then analysed by the *analyse* component that is responsible for determining the current state of the managed element. Based on this state, the *plan* component decides if any adaptation of the managed element's current configurations is needed. In case it is, it determines the set of adaptation actions to perform, that are then translated into a set of configurations and enforce by the *execute* component.

From an architectural perspective, autonomic computing systems are represented as an interactive collections of autonomic elements, each performing a specific operational function [7]. Each autonomic element consists of one or more managed elements that execute the operational function and an autonomic manager that controls the configuration, inputs and outputs of the managed elements according to the MAPE-K control loop model. The autonomic manager is also used as the interface to interact with the other autonomic elements and with human operators. Generic implementations of the architecture of an autonomic element have been proposed in [8][9].

### 2.1.3 From autonomic computing to autonomic networking

To overcome the drawbacks/limitations of current management approaches, recent research efforts have been focusing on applying and extending the autonomic computing

principles to network management systems, i.e. to the collective self-management of networks of individual computing devices [10][11][12]. More specifically, these efforts have been investigated how to apply self-management concepts and design to the various issues in the area of network management, such as fault, security, configuration, performance etc. [12]. According to the autonomic principles, network management systems are enhanced with self-aware, self-adaptive and self-optimising functionality, which is embedded within the network devices themselves. This allows the system to analyse and understand its environment, to decide upon the appropriate configuration actions, and to learn from previous decisions based on closed-loop feedback control solutions. In order to perform the control-loop functionality, sophisticated monitoring mechanisms are in charge of gathering network information, which can be used by decision making processes distributed across network nodes. In 2003 Clark et al. proposed in [13] a new paradigm for networks in the form of a Knowledge Plane, that the authors defined as a p̈ervasive system within the network that builds and maintains high-level models of what the network is supposed to do, in order to provide services and advice to other elements of the network.̈ They envisioned the network as an intelligent (cognitive) system that can configure and operate itself given high-level instructions.

Over the past few years, there have been some proposals for high-level autonomic network management architectures, e.g. [10][14][15][16], that have been investigated the self-management challenges from functional perspectives.

In fact several survey works have highlighted the variety of research challenges and efforts towards the autonomic paradigm [11][17][12][18] and have attempted to categorise the different works. In particular the authors In [18] argue that given the scope of research challenges involved in the design of network self-management systems, it is essential to investigate the main requirements of these systems. They propose some qualitative evaluation factors to compare the different systems. These factors evaluate especially the degree of reactivity/pro-activity of a system, its degree of adaptability, its degree of intelligence (i.e. its ability to learn), its degree of awareness (i.e. knowledge about the environment), the usage of historical knowledge and the degree of autonomy and autonomicity.

Recently the authors in [6] argue that evaluation methods for autonomic systems need to be refine and extended. They propose a new evaluation approach that relies on both quantitative and qualitative evaluation parameters. In particular they define a quantitative index to evaluate the learning ability of an approach and propose a method to measure the degree of awareness. In addition they define several cost functions to evaluate the cost incurred by the approach, especially in terms of computation and communication overhead. They finally propose strategies to evaluate the granularity of intelligence and the level of security provided.

## 2.2 Adaptive Network Resource Management

### 2.2.1 Introduction

Network resource management is performed through traffic engineering (TE) functionality that aims to control and optimise routing and traffic allocation functions in order to avoid the emergence of congestion hotspot [1][19]. Today's practices for managing network resources mainly rely on off-line TE approaches where the expected traffic demand is calculated from previous usage and used to produce a specific routing configuration, aiming to balance the traffic and optimise resource usage over long timescales, e.g. weekly or monthly [20]. These routing configurations are typically computed by a central management system based on the estimation of the traffic demand for the next provisioning period.

Off-line TE schemes have been extensively investigated both in the context of Multi Protocol Label Switching (MPLS)-based TE by setting the Label Switched Paths (LSPs) and in the context of IP-based TE by determining heuristics to tune the link weights that optimise some objective function given a set of traffic matrices [21][22][23][24][25]. However network traffic is known to be highly dynamic due to variation in user demands, link failures, BGP rerouting or flash crowds [26]. Given their static nature, these off-line approaches are enable to deal with the dynamics of traffic and network conditions that cannot be reflected in the demands estimation and as such, can be well sub-optimal in the face of changing or unpredicted traffic demand.

To overcome these limitations, online TE approaches able to react and adapt to current conditions on short-time scale have been proposed. The distinction between off-line and online approaches relies on the availability of traffic demands or the time-scale of operation. More specifically, in contrast to off-line TE schemes, online approaches do not rely on the knowledge of any traffic matrix to configure the routing or the link weights. Instead, they dynamically adapt the settings in short timescales in order to rapidly respond to traffic dynamics [20]. These schemes do not rely on any knowledge of future demands to configure the settings but instead use monitored real-time information from the network. As such, they aim at adaptively distributing the traffic load as evenly as possible onto the network according to the changing traffic conditions in order to accommodate unanticipated demands or traffic variations.

There have been some proposals for both online MPLS-based TE such as [27][28] and online IP-based TE such as [29][26][30]. These approaches focus on dynamically adjusting the volume of traffic (represented by splitting ratio) sent across several available paths between each source-destination pair in the network according to real-time traffic information.

### 2.2.2 Online Traffic Engineering

#### MPLS-based Solutions

Adjusting the volume of traffic to send across several pre-computed paths have been initially proposed in [27][28]. Based on periodical statistic information received from the network about the different paths, ingress routers dynamically modify the traffic splitting ratios across the set of available paths to the destination in order to optimise a given objective function.

**MATE** Ingress routers in MATE [27] use instantaneous knowledge of the network state to determine a new configuration that minimises total network cost, for which a simple gradient descent method is proposed. Each ingress-egress pair of nodes is associated with an agent that implements functionalities for collecting network information and load-balancing decisions. These decisions rely on an oracle system [28] that has a global knowledge of the network conditions and the different actions. However it is unclear how the consistency between the different load-balancing decisions is guaranteed.

**TEXCP** In [28] the authors propose TeXCP, an online traffic engineering approach that aims at minimising the maximum utilisation in the network. In TeXCP each pair of ingress-egress nodes is associated with a specific TeXCP agent that is implemented in the relevant ingress node. The TeXCP agents use information received from the egress nodes to assess the quality of each path (in terms of utilisation) and adjust the traffic splitting ratios so that to satisfy the optimisation objective. The adjustment actions are taken and enforced at network edges by each TeXCP agent without requiring any

explicit coordination with the other agents. However, in order to guarantee the consistency between the different adjustment decisions, the actual decision-making process is supported by a control mechanism implemented by the core nodes. These are responsible for monitoring the available bandwidth on their outgoing links and for informing the TeXCP agents that control some traffic flows over these links about the share of available bandwidth each of them can use. From an implementation perspective, the main advantage of TeXCP is that ingress nodes do not need to communicate among themselves to decide on the adjustment to perform. However, in order to guarantee consistencies between the different adjustment actions, an explicit control mechanism needs to deployed over all the network nodes, which involves communication between core and edge nodes.

### IP-based online TE solutions

**AMP** Unlike MPLS-based TE approaches such as MATE and TeXCP where reconfigurations are performed at ingress nodes only, the authors in [29] propose AMP (the Adaptive Multi-Path algorithm) - an online TE approach designed for IP-network where all nodes in the network are responsible for dynamically adjusting the traffic splitting ratios between the different available next hops, based on information received from upstream routers.

The main principle of AMP is drawn from [31], a early proposal to extend the Open Shortest Path First (OSPF) [32] routing protocol with adaptive traffic engineering (i.e. load-balancing) capabilities. In [31] network-wide link load information is used by the nodes to redistribute the traffic load towards less loaded network parts. The link load information is distributed through a network-wide flooding mechanism so that each node is responsible for advertising all the other nodes in the network about its outgoing links information. Based on the received information, a load-balancing algorithm implemented in each node identifies the highest loaded link in the downstream part of the network and adjusts the volume of traffic across the different next hops so that the traffic load is diverted towards less loaded downstream paths. This approach shows severe limitations in terms of implementation, especially given scalability issues associated with the network-wide flooding mechanism. In addition, in order to support the proposed scheme, complex and costly information storage structures need to be implemented in each individual network node.

In order to overcome these limitations, the authors in [29] propose a new approach to disseminate load-information. Unlike OSPF-OMP where each node has the same network-wide information about the network state, the nodes in AMP use only information received from direct downstream neighbours to take some load-balancing decisions. As such, an increase in the load of a link $l$ results in a situation where only the directly connected node immediately reacts to the increase by adjusting its splitting ratios across the different next-hops. To decide on the adjustment factors to apply, each node uses both local information and information received from the downstream next-hop nodes about the contribution of this node to the load of the downstream links. However, the approach does not guarantee that the consistency between the different decisions can always be satisfied.

**REPLEX** In [26], the authors propose REPLEX, another distributed online traffic engineering approach not restricted to path-centred networks, where a similar load information dissemination mechanism is used. Unlike in AMP where information about the contribution to the load of upstream links is not differentiated per destination, the signalling mechanism used in REPLEX is designed so that the contribution is reported per destination. Each network node is equipped with several REPLEX agents, each

one being responsible for a specific destination, so that load-balancing decisions are performed on a per destination basis. In order to decide on the splitting ratio adjustments, each agent a uses both local information and aggregated feedback information received from direct downstream neighbours only. Compared to AMP, REPLEX can perform a finer control of traffic load distribution since each agent can monitor the effects of splitting ratio setting decisions for a given destination. In order to guarantee consistent decisions between the different agents, the load-balancing algorithm relies on game-theoretic fundamentals.

Although a hop-by-hop information dissemination mechanism scales better than a network-wide flooding mechanism, the signalling communication overhead incurred by these distributed approaches is still significant since all nodes need to communicate to exchange information about the current state of the network in order to take reconfiguration decisions. In addition although simpler structures need to be implemented to maintain the traffic engineering information and decisions, these need to be deployed in all the network nodes.

**HOMEOSTATIS** To avoid flooding the network with signalling messages, the authors in [33] propose a scheme by which nodes use only local information from their direct outgoing links to decide whether or not to use them to route traffic. Each node in the network is provided with a set of available paths towards any destination and is responsible for selecting the number of paths to effectively use given the utilisation of its outgoing links. The paths are sorted according to their propagation delay such as preferentially paths with low delay are used to route traffic. Due to the local scope of information regarding network conditions, this approach does not target optimality but robustness. However, the main drawback of approaches focusing on robustness is that they often have poor performance in terms of resource utilisation in case of lightly loaded conditions in the network [34]. In addition, since decisions are based only on a local view of the global network state, independent decisions may not always be consistent with each other and as such, may lead to undesirable situations/configurations.

### Multi-topology routing for TE

Over recent years, there has been interest for the concept of logical/virtual planes, whereby a physical network infrastructure can be logically represented into several logical/virtual planes, each of which having its own configuration.

Multi-topology routing [35][36] is a standardised extension to the common Interior Gateway routing Protocols (IGP), i.e. Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (IS-IS), that aims at determining several independent virtual IP topologies based on a single network topology, each having its own independent routing configurations, especially its own link weight settings. Based on these link weight settings, the Shortest Path First (SPF) algorithm can be applied independently between each source-destination pair in each topology. Thus, it is possible to compute a set of paths between each pair of end points in the network with each path being related to one virtual topology. In order to determine according to which topology traffic packets need to be routed, these are marked at each source node with the identifier MT-ID of the routing topology the traffic has been assigned to. At the router level, a separate routing table needs to be produced for each routing scheme so that, upon receiving traffic packets, a router in the network analyses the MT-ID marked in the packets and forwards the packets to the next-hop according to the relevant routing table [37].

While multi-topology extensions have been initially developed for the purpose of routing different types of services and traffic across different paths in the network, there

has been interest to use these principles for other purposes such as network resilience [38][39][37] and intra-domain traffic engineering [40][41][30].

Using virtual topologies to support traffic engineering functionality has been investigated in particular in [40] where a new approach for intra-domain off-line traffic engineering in OSPF networks is proposed. Based on the configuration of $n$ virtual topologies, the traffic demand between any source-destination pair in the network is virtually partitioned into $n$ independent sets at ingress nodes, so that each traffic set is assigned to one of the $n$ topologies and routed according to that topology's configuration. The volume of traffic assigned to each virtual topology is determined according to certain arbitrary splitting ratios, computed in order to optimise the network resource usage by balancing the traffic load more evenly in the network.

In fact the incentives for employing multi-topology principles to support traffic engineering functionality rely on the current nature of IGP-based routing protocols that show some limitations to perform optimal resource distribution. In IP-based network, traffic engineering mainly relies on the equal cost multi-path (ECMP) [1] feature which allows to distribute the traffic only equally among multiple next hops of the equal cost paths between a source and a destination, and link weights manipulations. Compared to MPLS solutions where the traffic between any source-destination pair can be split arbitrarily across the paths, the traffic distribution is less flexible in IP-based networks, especially due to the restriction of the ECMP feature that authorises even traffic splitting only. Based on the features provided by using several topologies, it is thus possible to overcome the limitations of current IP-based traffic engineering practices. By exploiting the path diversity enabled by the configurations of the different topologies, a finer control granularity of the traffic distribution in the network can be performed through arbitrary traffic splitting and flexible load-balancing [40].

The incentives for using virtual topologies to support traffic engineering functionality have been further advocated in [41], where the authors argue that multi-topology routing should be extended for traffic engineering purposes. The author do not propose a detailed traffic engineering solution, but instead they discuss different models using multi-topology principles for off-line and online traffic engineering. In particular, they propose a method to compute the different logical topologies that relies on the authors' previous work where multi-topology principles were used for resilience purposes.

**AMPLE** Most recently, multi-topology routing infrastructure has been used in [30], where an adaptive traffic engineering approach targeting IP-networks is proposed. The approach (called AMPLE) relies on two components: an off-line link weights computation algorithm to configure the different topologies and an online centralised adaptation algorithm to dynamically adjust the splitting ratios of traffic into the different routing topologies according to network conditions. Unlike the aforementioned distributed approaches, a central controller that has a global knowledge of the network state is used in AMPLE to perform the re-configurations. The advantage of such a centralised decision-making process is that the consistency between different re-configuration actions is guaranteed. However, using a centralised approach has some limitations. First it is less scalable than a solution where decisions are taken by nodes themselves inside the network, since at each re-configuration period the central controller needs to gather information from all the links and nodes in the network, which incurs a significant communication overhead at between the central manager and devices. In addition, a lag in the central manager reactions may be introduced, which may result in suboptimal performance. It also presents some limitations in terms of reliability since it introduces a single of point of failure.

One of the challenging question when implementing a solution relying on multi-

topology routing relates to the actual number of virtual planes required to perform traffic engineering. Since routing information needs to be maintained for each topology, this makes the size of current routing table in routers to be multiplied by the actual number of topologies. If the number of topologies required is significant, these solutions may not be applied in practice, given the cost for storing the routing information in each router. Results in the approaches described above show that only a small number of topologies (typically between 3 and 5) is enough to offer pretty good path diversity. In particular the authors in [40] show that only a small number of traffic sets (2 or 4) is enough to achieve near-optimal performance.

**SculpTE** In parallel of the solutions focussing on computing the splitting ratios, the authors in [42] have been interested in an approach that dynamically tunes the link weights in reaction to load variation on the links. Based on information reported from the network, the proposed scheme modifies the link weight settings sequentially in only one of the different routing topologies at a time. Compared to other approaches, the authors do not elaborate neither on the requirements to configure the different topologies, nor on the number of topologies to use. In addition, it has been recently argued [43] that changing link weights too often should be avoided since this may cause instability and service degradation during the re-convergence process.

# Chapter 3

# DACoRM: a Decentralised, Coordinated and Adaptive Resource Management Scheme

## 3.1 Notations

This section introduces the main notations, definitions and parameters used in this report.

**Network topology** A network topology $T$ is represented as a graph $G = (V, L)$, with a set of vertices (nodes) $V$ and a set of links $L$. Any link is generically noted $l$. If for distinction purposes, it is necessary to explicitly introduced the two extremity nodes of a link, for instance the link between node $A$ and node $B$, the following extended notation is adopted, $l_{AB}$. Each link $l$ in $L$ is associated with a weight $w(l)$ and a capacity $c(l)$.

**Edge nodes and core nodes** Unless otherwise stated, a network is referred in this thesis as an intra-domain network. As such, the network nodes are classified into the set of edge nodes (i.e. nodes where traffic enters and exits the domain) and the set of core nodes (i.e. the other nodes, that simply forward traffic towards other nodes in the domain).

**Source nodes and destination nodes** The edge nodes are then themselves classified into the set of source nodes $\Sigma$ (i.e. the nodes where traffic enter the network) and the set of destination nodes $\Omega$ (i.e. the destination nodes of the incoming traffic demand). It is worth mentioning that it is possible for a source node to be also a destination node. In a Point-of-Presence (PoP)-level topology, for instance, each node can be either a potential source of traffic and destination node, therefore, $\Sigma = \Omega$. The number of elements in $\Sigma$ and in $\Omega$ is represented by $|\Sigma|$ and $|\Omega|$, respectively.

**Traffic flow** For any $i \leq |\Sigma|$ and $j \leq |\Omega|$, $(S_i - D_j)$ represents the source-destination pair of source node $S_i$ and destination node $D_j$. Any source-destination pair $(S_i - D_j)$ is associated with a volume of traffic $v(S_i - D_j)$ representing the traffic demand between source node $S_i$ and destination node $D_j$, so that the pair $((S_i - D_j), (v(S_i - D_j))$ is defined as the traffic flow associated to the source-destination pair $(S_i - D_j)$. The so-defined traffic flow is noted $F(S_i - D_j)$. Let $\phi_{S_i}$ be the set of traffic flows locally originating at source node $S_i$, i.e. those traffic flows for which $S_i$ is the source node:

$$\phi_{S_i} = \{\ F(S_i - D_j), \quad j \leq |\Omega|\ \}$$

**Link utilisation** For any link $l$ in the network, the utilisation of $l$, noted $u(l)$, is defined as the ratio of the load $\rho(l)$ actually traversing the link $l$ and the capacity $c(l)$ of the link $l$, i.e. $u(l) = \frac{\rho(l)}{c(l)}$.

**Maximum utilisation** The maximum utilisation in the network is represented by $u_{max}$, defined so that:

$$u_{max} = \max_{l \in L}(u(l))$$

The link with the maximum utilisation is noted $l_{max}$. If more than one link has the maximum utilisation, the extended link notation described previously is used to differentitate between the different links.

## 3.2 Overview of DACoRM

Despite recent proposals to enable adaptive traffic engineering in plain IP networks [29][26][30], network resource management normally relies on a *centralised* traffic engineering manager that periodically computes new configurations according to dynamic traffic behaviours. In order to meet the requirements of emerging services, network resource management functionality that is decentralised, flexible, reactive and adaptive to traffic and network dynamics is necessary.

This chapter introduces DACoRM (**D**ecentralised, **A**daptive, **Co**ordinated **R**esource **M**anagement), a new intra-domain resource management approach for IP networks, in which the traffic distribution is controlled in an adaptive and decentralised manner according to the network conditions. This adaptive control is performed at network edges, by the source nodes themselves that are equipped with adaptive traffic engineering capabilities that allow them to automatically re-optimise the distribution of user traffics into the network without the intervention of network administrators.

Compared to previous approaches where core nodes participate in the re-configuration process [29][26], adaptations in DACoRM are performed at network edges only. However, unlike [28], where the adjustment actions are supported by a control mechanism implemented by the core nodes, re-configurations in the proposed approach are the result of a coordination process that takes place between the source nodes without requiring neither any support from the other nodes, nor any modification of the core node functionalities. As the approach proposed in [30], DACoRM relies on multi-topology routing to provide the path diversity between any source-destination pair in the network. However, the resource management approach followed in DACoRM clearly differs from [30] given that it does not rely on any external and centralised manager to control the traffic distribution.

The rest of this section focuses on presenting and describing the main features of DACoRM.

## 3.3 Path diversity through Multi-Topology Routing

In a similar fashion to other traffic engineering schemes, DACoRM uses multi-topology routing [35][36] as the underlying network routing protocol to provide a set of available routes between each pair of edge nodes. A routing infrastructure using the multi-topology principles can provide the required flexibility to control the distribution of traffic in the network [40]. This section presents the requirements and a method to configure different virtual topologies in order to provide path diversity.

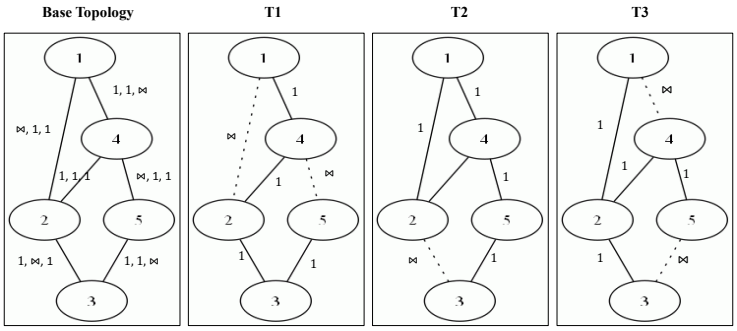


Figure 3.1: Building multiple topologies

### 3.3.1 Path diversity requirements

The configuration of the different virtual planes is part of an off-line process which computes a set of desired IP virtual topologies given the physical network topology. The derived topologies are such that two objectives are satisfied:

1. Providing a set of non-completely overlapping paths between the source-destination pairs in the network, i.e. there is always at least one path which is not overlapping with the others.

2. Avoid introducing critical links, i.e. given a link $l$ that is traversed by some traffic from node S to node D, there always exits an alternative path that can be used for routing the traffic without traversing $l$.

The idea of obtaining topologies that satisfy these requirements is the following. Assume that $l$ gets congested. The idea of using multiple paths is to be able to move some traffic away from this link towards other parts of the network, i.e. towards other links. Computing topologies which satisfy the above requirements ensures that for any link $l$ in the network, it is always possible to find at least one $(S - D)$ traffic flow that is routed over link $l$ in a set of topologies while it does not traverse $l$ in the set of those other topologies.

Figure (3.1) illustrates a simple example of how virtual topologies that satisfy the aforementioned requirements can be derived from a base physical topology. In this example, traffic between the source-destination pair 1-3 is forwarded from source node 1 towards destination node 3. In each of the alternative topologies $T_1$, $T_2$ and $T_3$, some links are assigned a *MAXIMUM* weight (represented here with infinity) which prevents these links from being used for routing the traffic demand between node 1 and node 3. With these settings, three non-overlapping paths can be determined between node 1 and node 3: (1;4;2;3), (1;4;5;3) and (1;2;3) and no critical link is created. The configuration of the alternative topologies is represented at the network level by associating a vector of link weights to each link in the network, each component of the vector being related to one topology. In this simple example only three virtual topologies are required to satisfy the objectives.

Obtaining the desired virtual topologies in more complex and realistic topologies, where each source-destination pair has to be taken into account, is not straightforward. The next section presents an algorithm to compute logical topologies according to the aforementioned requirements. The algorithm does not rely on any a priori knowledge of traffic demands to determine the different topologies. It uses only the characteristics of the physical network topology to derive the set of virtual planes.

It should be noted that balancing the traffic over the different paths provided by multi-topology routing may lead to route traffic over paths with longer round trip times. This may be an issue when considering traffic demands with certain level of quality of service. In this thesis it is assumed that only best effort traffic is targeted by the proposed scheme and as such, this is not considered as an issue in this work.

### 3.3.2 Virtual Topologies Computation Algorithm

The proposed algorithm to compute the virtual topologies relies on the method described in [41]. The idea of this method is to create $n$ several copies $T_k, k \in [1, n]$, from the original physical topology $T_{base}$ and to remove a subset of links in each of the logical topology $T_k$ in a round-robin manner, such as links are removed in such a way that each logical topology is still connected, i.e. all nodes are reachable in all topologies. The pseudo-code of the algorithm is presented in Algorithm (3.3.1).

Initially the algorithm sorts all the links in decreasing order of the inverse of their capacity. It then iterates through all the links such as links with the smallest capacity are considered first. For each link, the algorithm tries to remove it from one of the topologies. A link can be removed from a topology if removing the link does not disconnect the topology. There exists some links cannot be removed from any topology without causing a network disconnection. This is especially the case of links associated to a node with a degree of connectivity equals to one. In this case, the node is indeed connected to the rest of the network through only a single link. The algorithm disregards all those links by analysing the connectivity of the nodes associated to each link. If one of the node has a connectivity equals to one, then the link is considered as non-removable and is not tested by the algorithm.

For each link, the algorithm tries to remove it from one of the topology $T_k$. If the link cannot be removed from $T_k$, then the algorithm tries sequentially the $T_{(k\%n)+1}$ topology until all topologies are analysed. The first topology analysed is chosen differently at each iteration (i.e. for each link to remove) such as the number of links removed is almost the same in each topology. The algorithm stops when exactly all the links are removed from exactly one logical topology or if a link cannot be removed from any of the logical topologies (i.e. removing the link disconnects the topology).

The reason for links to be considered in the decreasing order of the inverse of their capacity is the following. Since links with the smallest capacities are more likely to become highly loaded, it is more likely that traffic needs to be diverted away from these links and as such, it is important that there exists alternative topologies not using them. In case where the number of topologies to use is fixed a priori to a low value that does not permit to remove all the links from $T_{base}$, the algorithm can therefore ensure that the path requirements are satisfied for the most sensitive links.

The computed topologies are then used to build the vector of link weights assigned to each link in the physical network, such as links removed in topology are represented by the *MAXIMUM* value in the vector of link weights while the weight of the other links is kept to its initial value in the physical base topology. In [41] the authors argue that less than six logical topologies are typically needed to cover all links with this method.

## 3.4 Adaptive Resource Management

### 3.4.1 Principles

Based on the path diversity provided by configuring the different virtual topologies, an adaptive resource management algorithm is responsible for re-optimising the utilisation

**Algorithm 3.3.1** Pseudo-code to compute the logical topologies

**Notations**

$L$ set of links in the network

$n$ number of desired virtual topologies

**Pseudo-code**

```
Create n copies T_k of the physical topology
Sort links l_i in L by decreasing order of the inverse of their capacity
i ← 0
while allLinksAnalysed = false and connected = true do
  nbCheckTopo = 0
  k ← 0
  while findTopo = false do
    Remove l_i in T_k
    nbCheckTopo++
    if T_k still connected then
      findTopo = true
      i++
      k ← k%n + 1
    else
      k ← k%n + 1
    end if
    if nbCheckTopo = n then
      connected = false
    end if
  end while
end while
```

of network resources in reaction to traffic dynamics. The algorithm controls the distribution of traffic load in the network in an adaptive and decentralised manner through re-configuration actions that dynamically adjust the splitting ratios of some traffic flows (as defined in Section 3.1) so that the traffic is periodically re-balanced from the most utilised links towards less loaded parts of the network. Only source nodes participate in the adaptation process.

The traffic demand between each source-destination pair is divided into $n$ sets at source nodes, with each set being associated to one of the $n$ topologies $T_k$. The proportion of traffic assigned to each set is determined by splitting ratios $x_{T_k}^{S-D}$, so that the two following conditions are satisfied:

$$\forall k \in [0; n], \quad 0 \leq x_{T_k}^{S-D} \leq 1 \tag{3.1}$$

$$\sum_{k=0}^{n} x_{T_k}^{S-D} = 1 \tag{3.2}$$

These are used to distribute incoming packets at source nodes and packets are routed to their destination according to the configuration of the topology they have been assigned to. Splitting ratios are not pre-computed by an off-line process as in other approaches, e.g. [40], but are instead adapted dynamically by the source nodes themselves, even without centralised control as is the case in [30]. The new splitting ratios are computed by a re-configuration algorithm that executes only at source nodes which allows them to react to traffic dynamics in an online fashion by adjusting the proportion of traffic assigned to each topology based on network conditions. Splitting ratios are decided by source nodes only and are not modified by other nodes on the route (i.e. no further splitting is permitted along the path).

The objective of this adaptive control is to permanently minimise the utilisation of the most utilised link in the network. Minimising the maximum utilisation in the network is a common objective considered by many load-balancing/traffic-engineering schemes in the literature (e.g. [23][24][25][28]). This allows to reduce the utilisation of hotspots against dynamic traffic behaviours, and as such reduces congestion risks by spreading the load over available paths [19]. Adaptively distributing the traffic load as evenly as possible onto the network also permits to ensure that future traffic demands can be satisfied.

Performing an edge-based control of traffic provides several advantages. It offers a better scalability since source nodes only need to be involved in the adaptive process and no modification of core nodes is required. In addition, edge nodes have detailed information about the incoming TCP flows, so they can perform a finer-grained control on packets distribution that follows the flow boundaries [40]. Packets belonging to the same TCP flow are always assigned to the same topology and no further adjustment is permitted along the route. This ensures that all packets in one flow follows only a single path to the destination, and as such, avoid introducing packet out-of-delivery problems that deteriorates TCP performance [44].

The adaptation process is performed in short-time scales, for instance, in the order of 5-10 minutes, which is in accordance with the common network monitoring interval [45][30].

### 3.4.2 In-Network Overlay of Source Nodes

Performing a re-configuration involves adjusting the traffic splitting ratios for some of the source-destination pairs for which traffic is routed across the link with the maximum utilisation in the network, $l_{max}$. This means that more traffic is assigned to topologies

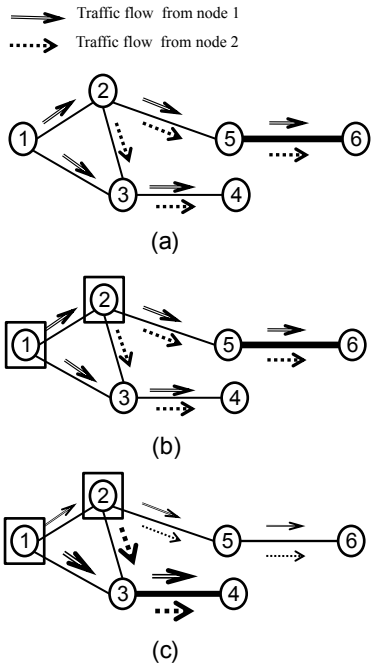


Figure 3.2: Example of conflicting decisions between node N1 and node N2

not using $l_{max}$ to route traffic thus decreasing the traffic volume assigned to topologies that do use $l_{max}$.

In realistic scenarios, links in the network are traversed by multiple traffic flows (i.e. are used by several source-destination pairs to route the corresponding traffic demand) and therefore, several source nodes may be eligible to adapt the ratios of traffic flows traversing $l_{max}$. Due to the limited network view of individual source nodes, actions taken by more than one node at a time may lead to inconsistent decisions, which may jeopardise the stability and the convergence of the overall network behaviour [30]. For instance, in the process of shifting traffic away from $l_{max}$, the different reacting nodes can re-direct traffic flows towards the same links, as depicted in Figure (3.2), thus potentially causing congestion. In Figure 3, source nodes $N_1$ and $N_2$ send traffic over link $l_{5-6}$. In (a), $l_{5-6}$ is identified as being the most utilised link in the network. $N_1$ and $N_2$ both react to this information (b) and decide to perform some re-configurations locally, which results in more traffic from both $N_1$ and $N_2$ routed towards link $l_{3-4}$, which then becomes overloaded (c).

To avoid such inconsistent decisions, the re-configuration actions to perform are decided in a coordinated fashion between a set of source nodes. In order to support this coordinated decision-making process, the source nodes are organised into an *in-network overlay* (INO), where the relevant entities can exchange information about the course of actions to follow in order to re-balance the network load. Overlay networks have received a lot of attention from the research community over the last decade, especially in the context of peer-to-peer networks [46][47]. An overlay network can be defined as a virtual network of nodes and logical links built on top of an existing physical network. In this work the INO refers to the dedicated logical plane/infrastructure to support interactions between the set of *adaptive traffic engineering* capable nodes in the process of deciding upon the re-configuration actions to perform in the network. The INO is used solely for the signalling, i.e. in-network management, between source nodes for coordination purposes, but not for direct traffic routing/forwarding.

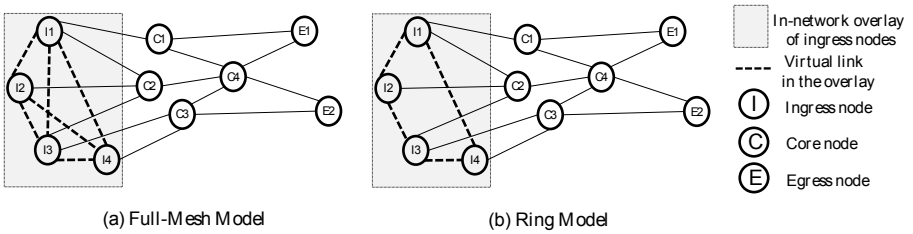


Figure 3.3: Example of a network and its associated in-network overlay of ingress nodes

The INO of source nodes is built during the initial configuration of the network in an off-line manner. Its formation is based on the identification of ingress nodes in the physical network, i.e. the nodes which are potential sources of traffic. In the case of a PoP (Point of Presence) level network, for instance, each node is a potential source of traffic and would therefore be part of the INO. Different topologies can be used to connect the nodes in the INO, e.g. a full-mesh or a ring. Such topologies are depicted in Figure (3.3), where the four identified ingress nodes on the physical network are logically connected in a full-mesh (a) and in a ring (b).

Each node in the INO is associated with a set of neighbours, i.e. nodes that are directly connected to the INO. As it can observed in the figure, all nodes are neighbour to each other in the full-mesh model since there exists a direct virtual link between all source nodes. In the ring topology, an INO node is only connected to two other nodes and therefore has only two neighbours. From a topological perspective, the full-mesh topology offers a greater flexibility in the choice of neighbours with which to communicate since all source nodes belong to the set of neighbours. However the choice of the INO topology may be driven by different parameters related to the physical network, such as its topology, the number of source nodes, but also by the constraints of the coordination mechanism and the associated communication protocol. The number and frequency of messages exchanged, for example, are factors that influence the choice of topology.

The structure of the different topologies and the associated communication model to support interaction between the nodes in the INO are described in details in Chapter 5 of this report.

## 3.5 System design

In order to realise the proposed adaptive resource management scheme, source nodes are equipped with *adaptive traffic engineering* capabilities. This requires that a set of components need to be deployed at source nodes as depicted in Figure (3.4).

Each source node $S_i$ maintains locally both static and dynamic information related to each of the traffic flows in $\phi_{S_i}$. This information is stored in two tables called the Link Information Table (LIT) and the Demand Information Table (DIT) respectively - Tables (3.1) and (3.2) shows the structure of an entry of each table. The LIT contains static information about the links traversed by the paths of all the traffic flows in $\phi_{S_i}$. It has an entry for each of the links in the network that is used by at least one the $(S_i - D_j)$ pairs in $\phi_{S_i}$ to route the corresponding traffic demand in at least one of the topology $T_k$. Each entry contains the identifier of the link $l_m$, the link capacity $c(l_m)$, references to the $(S_i - D_j)$ pairs in $\phi_{S_i}$ that use that link for routing their associated traffic flow in at least one topology, and for each of these $(S_i - D_j)$ pairs, the involved topology(ies). Based on this information, source nodes can efficiently determine if a traffic flow contributes to

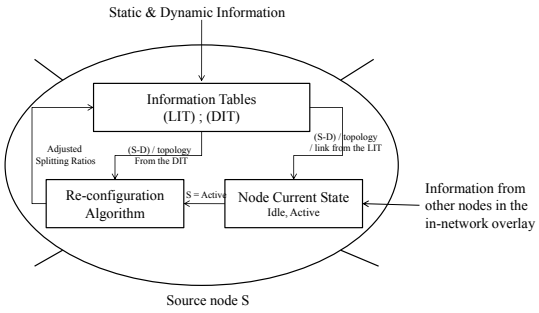


Figure 3.4: Components overview at the source node level

| | | | |
|---|---|---|---|
| $l_m$ | $c(l_m)$ | $(S_i - D_1)$ | $T_1$ |
| | | | ... |
| | | | $T_n$ |
| | | ... | ... |
| | | $(S_i - D_N)$ | $T_1$ |
| | | | ... |
| | | | $T_n$ |

Table 3.1: Entry of the Link Information Table (LIT)

the load of a link in the network, and if so, on which topology(ies). The DIT contains dynamic information related to each of the traffic flows in $\phi_{S_i}$. It maintains the current and previous splitting ratios assigned to each topology, respectively $x_{T_k}^{cur}(S_i - D_j)$ and $x_{T_k}^{prev}(S_i - D_j)$, as well as the associated traffic volume $v(S_i - D_j)$ for the current time interval. This volume of traffic can be easily monitored locally by the relevant source node.

Based on information stored in the tables and on information received through the *in-network overlay*, a source node can determine its current state, which can be either idle or active depending on whether or not it needs to perform re-configuration. When in the latter state, the node executes the re-configuration algorithm over its locally originating flows to determine the new splitting ratios that can achieve the traffic engineering objective (i.e. decrease the utilisation of the most utilised link in the network). If with these new splitting ratios no other link in the network gets overloaded, the adjusted splitting ratios are updated in the corresponding DIT and enforced at the next time interval.

Network information is disseminated to source nodes thanks to traffic engineering capabilities of enhanced Interior Gateway Protocols, that can incorporate traffic engineering metrics into link state advertisement [48].

| | | | | |
|---|---|---|---|---|
| $(S_i - D_j)$ | $v(S_i - D_j)$ | $T_1$ | $x_{T_1}^{prev}(S_i - D_j)$ | $x_{T_1}^{cur}(S_i - D_j)$ |
| | | ... | ... | ... |
| | | $T_n$ | $x_{T_n}^{prev}(S_i - D_j)$ | $x_{T_n}^{cur}(S_i - D_j)$ |

Table 3.2: Entry of the Demand Information Table (DIT)

# Chapter 4

# DACoRM Adaptation Process

## 4.1 Principles

The overall objective of DACoRM is to balance the load in the network by moving some traffic away from highly utilised links towards less utilised ones in order to reduce the utilisation of the hot spots against dynamic traffic behaviours. To achieve this objective, the proposed adaptive resource management algorithm successively adjusts the splitting ratios of traffic flows through a sequence of re-configuration actions (or adjustments) that constitute the adaptation process. To prevent inconsistencies between concurrent traffic splitting adjustments, the adaptation process is designed so that re-configuration actions are performed sequentially: only one source node is permitted to perform a splitting ratio adjustment at a time (i.e. to change the splitting ratios of one of its local traffic flows). The overall load-balancing objective is thus achieved through a combination of successive adjustments.

At each sequence that constitutes the adaptation process, one source node $S_i$ in the in-network overlay is selected to compute new splitting ratios for one of its locally originating traffic flows in $\phi_{S_i}$ through a re-configuration algorithm, with the objective to move traffic away from the most utilised link in the network.

The re-configuration algorithm uses information from the network concerning the link with the maximum utilisation, $l_{max}$, and the set of other heavily utilised links, $S_{HU}$. The latter is defined as the set of links in the network with a utilisation within $\alpha\%$ of the utilisation of $l_{max}$ so that:

$$S_{HU} = \{l \in L/ \quad (1-\alpha) \cdot u(l_{max}) \leq u(l) \leq u(l_{max}), \quad \alpha \in [0;1]\}$$

Based on this information, the objective of the re-configuration algorithm is to try to modify the splitting ratios of those traffic flows in $\phi_{S_i}$, which contribute to the load on $l_{max}$ so that: a) some traffic is moved away from $l_{max}$, and, b) the diverted traffic is not directed towards links in the set $S_{HU}$ which are potentially vulnerable. A situation that should be avoided is that excessive traffic demands are diverted to a link which is originally not in $S_{HU}$, so that its new utilisation becomes higher than that in $S_{HU}$.

The definition of the set of highly utilised link is relative to the actual utilisation of $l_{max}$, and is affected by the value of the parameter $\alpha$. This can be between 0 and 1. The set $S_{HU}$ may include no other link in the network (this is especially the case with $\alpha = 0$) or all the links in the network (for instance if $\alpha = 1$). The choice of the value of the parameter $\alpha$ does not follow any specific rule. However, the performance of the adaptation process may be directly affected by its value. In particular, since the objective is to alleviate those links which are the most utilised, the value of $\alpha$ should be large enough to avoid diverting traffic towards links with a utilisation close to the maximum. At the same time, it should also be small enough to prevent a situation

where the flexibility to divert traffic towards other links is limited given that a large number of links are preserved from receiving diverted traffic. Typically the value of $\alpha$ is chosen to be between 0.05 and 0.1, representing a gap of 5% and 10%, respectively, from the maximum. The influence of the value of the parameter $\alpha$ is further extended and discussed in Section 4.5.

The adaptation process terminates if a successful configuration cannot be determined or if it reaches the maximum number of permitted iterations (a parameter of the adaptive resource management algorithm).

## 4.2 Selecting the Deciding Entity

### 4.2.1 Objective

At each sequence/iteration of the adaptation process, one source node in the *in-network overlay* - called the Deciding Entity (DE) - is selected to initiate re-configuration actions. The selected DE is responsible for executing the re-configuration algorithm over its locally originating traffic flows with the objective to re-balance the network load. The DE role can be taken by any node in the *in-network overlay*. In order to select a unique DE, a selection rule/logic needs therefore to be deployed within the different source nodes, so that each source node is able to determine independently whether or not it can assume the DE role for the new re-configuration interval. This logic is deployed in the C̈urrent State Componentäs depicted in Figure (3.4) in Section 3.5.

Since the objective of the re-configuration algorithm is to divert traffic from the link $l_{max}$ with the maximum utilisation in the network, a node should be selected to be the DE for the current re-configuration interval only if it is potentially able to divert some of its local traffic flows away from $l_{max}$. A node such that none of the locally originating traffic flows is routed over $l_{max}$, for instance, would not be able to divert traffic away from $l_{max}$ and as such should not be selected to be the DE.

As such, the idea to select the DE is to identify for each link $l$ in the network all the source nodes that use the link $l$ to route at least one of their local traffic flows in some topologies and to select one among these source nodes to be the DE if link $l$ becomes the most utilised link. In order to facilitate the overall process, this selection is not done at run time but instead relies on an off-line procedure that associates each link in the network to a single node in the *in-network overlay* (i.e. a source node). Since the number of links in the network can be larger than the number of source nodes, each source node is therefore associated to a list of links, so that as all the lists are disjoint. Upon receiving condition information about the link $l_{max}$, each source node checks whether $l_{max}$ falls within its associated list of links ($l_{max}$ can fall within only one list of links). If so, the relevant source node assumes the DE role for the new re-configuration interval.

### 4.2.2 Association Procedure

The association procedure works as follows. Initially, each source node $S$ is associated with the list of its outgoing links $O_S$ - the list of associated network links is noted $A_S$. The algorithm then considers one by one the other links in the network (i.e. core links) to determine to which source node these need to be associated. This decision relies on two metrics:

- the Node Contribution - noted $NC_S^l$ - of a source node $S$ to the core link $l$ currently analysed that evaluates the number of traffic flows in $\phi_{S_i}$ routed over the link $l$ in at least one topology but not all topologies. The value of $NC_S^l$ can be easily

obtained from the Link Information Table (Figure, Section 3.5) maintained at each source node.

- the size of the current associated list of network links $|A_S|$.

These two metrics are computed for each source node.

First the algorithm compares the $NC_S^l$ values of each source node $S$. Intuitively the highest is the number of local traffic flows that can be potentially diverted by a source node, the highest are the chances for this source node to be able to perform some adjustments/re-configurations. The link $l$ is therefore associated to the source node with the highest $NC_S^l$. If this metric is not enough to select a unique source node (several source nodes obtain the equal maximum value of $NC_S^l$), then the algorithm compares the $|A_S|$ value of all the source nodes S with the equal maximum $NC_S^l$ . In order to more evenly balance the size of the different $|A_S|$, the link $l$ is associated to the source node that obtains the minimum $|A_S|$ value. This is to avoid a situation where only a small subset of source nodes can be selected to be the DE. Since a source node can only modify the splitting ratios of its locally originating traffic flows, this may lead to a situation where the adaptation process output is constrained by the number of traffic flows that can be modified, which may result in sub-optimal performance. If this criteria is still not enough to determine a unique DE (i.e. the minimum is not obtained by a unique source node), the link $l$ is then randomly assigned to one of the source nodes with the minimal $|A_S|$.

The pseudo-code of the algorithm to associate the network links to the source nodes is shown in Algorithm (4.2.1).

## 4.3 Delegation Process Overview and Principles

While the Deciding Entity (DE) is initially selected to perform re-configuration actions, it may not always be able to determine by itself a configuration with which traffic can be shifted away from $l_{max}$ such that the utilisation of $l_{max}$ can be reduced while no other link in the network obtain a new utilisation higher than the original utilisation of $l_{max}$. In such a case the DE needs to delegate the re-configuration task to other nodes in the *in-network overlay*.

Upon failure to determine an acceptable configuration, the DE sends a delegation request to some of its neighbours in the *in-network overlay* through the overlay infrastructure. When receiving such a request, neighbouring nodes, called Selected Entities (SEs), execute the splitting ratio re-configuration algorithm independently. Their results are communicated back to the DE, which then selects the configuration to apply (among successful ones), and notifies the relevant SE to enforce their new splitting ratios. To compare the different successful re-configurations and select one of them, extra information about the re-configuration proposed by each SE is also provided to the DE. This indicates the contribution degree of the traffic flow for which adjustments are proposed to the load of $l_{max}$, i.e. the ratio of the total volume of the traffic flow currently routed over $l_{max}$ to the load of $l_{max}$. Using this information, the DE selects the solution with the highest contribution since this may result in off-loading more traffic from $l_{max}$ and as such, increase the convergence of the algorithm. To limit the number of messages exchanged and the response time, a delegation process can only be initiated by a DE.

Choosing the neighbours to which a delegation request is sent can influence the responsiveness and performance of the algorithm. Sending a request to only a limited number of neighbours can minimise the number of messages exchanged, as well as computation/communication overhead, but can also decrease the probability of discovering

**Algorithm 4.2.1** Pseudo-code for the 'Association Link - INO Node' Algorithm

**Notations**

$S$ a source node in the network
$O_S$ set of outgoing links of source node $S$
$C$ set of core links in the network (the incoming extremity is not a source node)
$A_S$ list of links assigned to source node $S$

**Pseudo-code**

```
for all nodes S do
    A_S ← all the links l in O_S
end for
for all links l in C do
    for all source nodes S do
        compute NC_S^l
        compute |A_S|
    end for
end for
max ← 0
min ← 0
for all links l in C do
    max ← maximum of the NC_S^l
    if one and only node S is such as NC_S^l = max then
        A_S ← l
    else
        min ← minimum |A_S| where S is such that NC_S^l = max
        if one and only node S is such as |A_S| = min then
            A_S ← l
        else
            randomly assign l to one of the node S that is such as NC_S^l = max and
            |A_S| = min
        end if
    end if
    update |A_S|
    reinitialise max and min to 0
end for
```

a node that can perform a successful re-configuration for further improvement of network performance. This trade-off can be parameterised by varying the number of SEs in the delegation process. It is also worth mentioning that to limit the extra communication overhead incurred by the delegation process, the reconfiguration decisions are taken locally by the DE as much as possible.

## 4.4 Re-configuration Algorithm

The re-configuration algorithm can be executed by source nodes only. Its objective is to determine if a new configuration (i.e. new splitting ratios) can be performed in order to move traffic away from the most utilised link in the network. It thus consists of finding one among the flows in $\phi_S$ for which some traffic can be diverted and for computing the associated adjusted splitting ratios. The algorithm consists of three phases that are performed according to the state of the node executing the algorithm, i.e. the node is acting either as the Deciding Entity (DE) or as a Selected Entity (SE). Without loss of generality, it is assumed that the re-configuration algorithm is executed by source node $S_i$.

### 4.4.1 Phase One: Local Adjustments

The objective of this phase is to determine if a re-configuration can be performed on one of the traffic flows in $\phi_{S_i}$. The outcome of this phase is either positive, which means that part of a local flow can be diverted from $l_{max}$, or negative if this is not possible. The execution of this phase does not depend on the state of the source node $S_i$.

The algorithm first identifies the local flows $F(S_i - D_j)$ that can be diverted from $l_{max}$. A flow qualifies if:

1. It is routed over $l_{max}$ in at least one topology.
2. It is not routed over $l_{max}$ in all topologies, i.e. there exists at least one alternative topology in which the traffic is not routed over $l_{max}$.

For each $F(S_i - D_j)$ that satisfies the two conditions, the algorithm defines two sets:

- $S_{l_{max}}^{S_i-D_j}$ the set of routing topologies that use $l_{max}$ to route $F(S_i - D_j)$
- $\bar{S}_{l_{max}}^{S_i-D_j}$ the set of routing topologies that do not use $l_{max}$ to route $F(S_i - D_j)$

The set $\bar{S}_{l_{max}}^{S_i-D_j}$ is then itself partitioned into two subsets: the set of topologies in $\bar{S}_{l_{max}}^{S_i-D_j}$ that can avoid using any link from $S_{HU}$ and the set topologies in $\bar{S}_{l_{max}}^{S_i-D_j}$ that use at least one link from $S_{HU}$. Based on these characteristics, the algorithm then classifies each $F(S_i - D_j)$ into two categories as follows:

- **Category I** - set of flows for which there exists at least one topology in $\bar{S}_{l_{max}}^{S_i-D_j}$ that do not use any link in $S_{HU}$
- **Category II** - set of flows for which all topologies in $\bar{S}_{l_{max}}^{S_i-D_j}$ are using at least one link in $S_{HU}$

The algorithm then considers each of the flows in Category I at a time and tries to adjust the splitting ratios. These are adjusted such that the ratios related to the topologies in $S_{l_{max}}^{S_i-D_j}$ are decreased while the ratios related to the topologies in $\bar{S}_{l_{max}}^{S_i-D_j}$ are increased.

The actual algorithm for adjusting the splitting ratios of a flow is presented in Section 4.5. The resulting configuration is then analysed to decide whether it is acceptable or not. A new configuration is said to be acceptable if:

1. the utilisation of $l_{max}$ is decreased
2. no link $l$ in the network attains a utilisation higher than the original value of $l_{max}$

This is represented by the two following equations:

$$u_{n+1}(l_{max}) < u_n(l_{max}) \tag{4.1}$$

$$\forall l \in L, \quad u_{n+1}(l) \leq u_n(l_{max}) \tag{4.2}$$

where $u_n(l)$ represents the utilisation of the link $l$ at the iteration $n$ of the adaptation process (i.e. the current iteration - before any new configuration is applied) and $u_{n+1}(l)$ represents the utilisation of the link $l$ at the iteration $n+1$ of the adaptation process (i.e. the utilisation obtained as the result of the enforcement of the new configuration).

Equation (4.1) indicates that the utilisation of the link identified as $l_{max}$ at the current iteration of the adaptation process is strictly decreased but does not necessarily imply that the maximum utilisation in the network is also strictly decreased. Since several links may have the same maximum utilisation, it is possible for the utilisation of the link identified as $l_{max}$ at the next iteration to be equal to the maximum utilisation in the network at the previous iteration, i.e. $u_{n+1}^{max} = u_n^{max}$.

These two conditions ensures, however, that the sequence of the maximum utilisation in the network at each iteration $(u_n^{max})_{n \in \aleph}$ is monotonically decreasing, i.e.

$$u_{n+1}^{max} \leq u_n^{max} \tag{4.3}$$

As such, the maximum utilisation in the network converges towards a lower value that depends on initial conditions.

If these conditions are satisfied, the new splitting ratios are accepted. The result of the algorithm is set to positive and the next iteration of the adaptation process (i.e. re-configuration action) is triggered. If none of the local flows can satisfy the requirements, the result of the first phase is set to negative and the algorithm enters the second phase.

### 4.4.2 Phase Two: Delegation

This phase is executed only if the source node $S_i$ is the DE. In case of unsuccessful local adjustments, the DE triggers a delegation process by sending a request to its neighbours in the in-network overlay for further attempts at alternative locations. Each neighbouring node in the in-network overlay is responsible for executing the first phase of the re-configuration algorithm on its local flows, the result of which is communicated back to the DE. The DE is then responsible for selecting one of the proposed new configurations among the positive results and for notifying the corresponding neighbour about the decision.

### 4.4.3 Phase Three:

This phase is executed only if the source node $S_i$ is the DE. If none of the neighbours is able to perform a re-configuration, i.e. all results are negative, the DE can resort to using links from the set $S_{HU}$. A traffic flow among the ones in Category II is randomly selected and its splitting ratios are adjusted. If no such flows can be identified, the result of the re-configuration algorithm is set to negative.

### 4.4.4 Time-complexity analysis

In order to be implemented, the re-configuration algorithm should be lightweight in terms of computational overhead (i.e. time-complexity) imposed at each source node. The time-complexity of the first and third phases of the algorithm is dominated by the number of locally-originated flows to consider, which depends on the size of the network. In the case of a Point of Presence level topology with $N$ nodes, for instance, where there are traffic demands between any pair of nodes in the network, this is $O(N-1)$. The actual cost of the second phase is related to the communication overhead. It has to select one solution among several ones and as such, its complexity depends on the complexity of the selection policies. Since these policies are lightweight in terms of computation (find the maximum solution), it requires negligible CPU computation. The overall time-complexity of the algorithm is therefore very low.

## 4.5 Adjustment of traffic splitting ratios

### 4.5.1 Objective of the adjustment algorithm

In order to determine if new configurations can be applied to a local traffic flow $F(S_i - D_j)$, the re-configuration algorithm relies on an *adjustment algorithm*. The objective of this algorithm is to determine whether or not any adjustment can be applied to the current splitting ratios $x_{T,cur}^{S_i-D_j}$ of $F(S_i - D_j)$ and if so, to compute the new splitting ratios $x_{T,new}^{S_i-D_j}$. In that case, the current splitting ratios are adjusted so that the ratios for the topologies in $S_{l_{max}}^{S_i-D_j}$ are decreased and the ratios for the topologies in $\bar{S}_{l_{max}}^{S_i-D_j}$ are increased according to some adjustment factors, respectively $\delta^-$ and $\delta^+$, that depend on the volume of diverted traffic from $l_{max}$, so that:

$$\forall T \in S_{l_{max}}^{S_i-D_j}, \quad x_{T,new}^{S_i-D_j} = x_{T,new}^{S_i-D_j} - \delta^- \tag{4.4}$$

$$\forall T \in \bar{S}_{l_{max}}^{S_i-D_j}, \quad x_{T,new}^{S_i-D_j} = x_{T,new}^{S_i-D_j} + \delta^+ \tag{4.5}$$

The new computing ratios $x_{T,new}^{S_i-D_j}$ need to satisfy the conditions (3.1) and (3.2) defined in Section 3.4.1, i.e.

$$\forall T_k, \quad 0 \leq x_{T_k,new}^{S_i-D_j} \leq 1$$

$$\sum_{T_k} x_{T_k,new}^{S_i-D_j} = 1$$

### 4.5.2 Computing the volume of traffic to divert

Determining the actual volume of traffic to be diverted from $l_{max}$ in each iteration is a challenging step/task. If too much traffic is shifted, other links in the network may become overloaded. This may cause oscillations as in the next iteration traffic will need to be removed from these links. In order to preserve the network stability, the volume of traffic that can be diverted in each iteration is therefore constrained by an upper bound $V_{max}$. More specifically, the upper bound $V_{max}$ is determined as follows.

**Notations** Without loss of generality, let suppose to be at the iteration $n$ of the adaptation process and $F(S_i - D_j)$ be the traffic flow for which adjustments are investigated. Let $l$ be a link in the network that is not included in the set $S_{HU}$, i.e. $l \in L \backslash S_{HU}$, and $u_n(l)$ be the current utilisation of link $l$ (i.e. before any new configuration is applied), so that $u_n(l) = \frac{\rho_n(l)}{c(l)}$, where $\rho_n(l)$ is the current load on link $l$ and $c(l)$ its capacity.

Let $u_n^{max}$ be the current maximum utilisation in the network and $l_{max}$ the link with the maximum utilisation (if several links have the same maximum utilisation, one among them is selected to be $l_{max}$). The utilisation of the link $l_{max}$ is so that, $u_n(l_{max}) = u_n^{max}$.

Finally let $u_{new}(l)$ represents the utilisation of link $l$ obtained as the result of the enforcement of new splitting ratios. In case these are selected and applied, the utilisation of link $l$ at the next iteration of the adaptation process is so that $u_{n+1}(l) = u_{new}(l)$.

**Upper bound $V_{max}$** According to the re-configuration algorithm objective, the splitting ratios of a traffic flow $F(S_i - D_j)$ can be adjusted if by doing so, none of the links in the network obtain a utilisation higher than the original utilisation of $l_{max}$. This means that for all links $l$ not including in the set $S_{HU}$, the following inequality must be satisfied:

$$\forall l \in L \backslash S_{HU} \quad u_{new}(l) < u_n(l_{max}) \tag{4.6}$$

Since the enforcement of new splitting ratios results in some traffic being moved towards other parts of the network, this may affect the load of a subset of links in the network. In particular, four cases can be considered:

- **Case 1**: the link $l$ is not used in any topologies to route the traffic flow $F(S_i - D_j)$. Its load is therefore not affected by the new configuration.
- **Case 2**: the link $l$ is used only in the topologies in the set $S_{l_{max}}^{S_i - D_j}$ to route the traffic flow $F(S_i - D_j)$. Since some traffic is moved away from these topologies, this results in a decrease in the load of link $l$.
- **Case 3**: the link $l$ is used only in the topologies in the set $\bar{S}_{l_{max}}^{S_i - D_j}$ to route the traffic flow $F(S_i - D_j)$. Since some traffic is moved towards these topologies, this results in an increase in the load of link $l$.
- **Case 4**: the link $l$ is used both in the topologies in the set $S_{l_{max}}^{S_i - D_j}$ and in the topologies in the set $\bar{S}_{l_{max}}^{S_i - D_j}$ to route the traffic flow $F(S_i - D_j)$. This situation can occur given that the paths provided through the configuration of the different virtual topologies are not necessarily completely disjoint (as indicated in Section 3.3.1). In this case, the variation of load of link $l$ depends on the difference between the volume of traffic diverted away and the volume of traffic received.

Noting $v$ the total volume of traffic diverted over link $l$ and $v^{'}$ the total volume of traffic moved away from link $l$, the new utilisation of $l$ can be obtained as follows:

$$u_{new}(l) = \frac{\rho_n(l) + v - v^{'}}{c(l)} \tag{4.7}$$

where the parameters $v$ and $v^{'}$ can be interpreted as follows:

$$Case\ 1 : v = 0\ and\ v^{'} = 0$$

$$Case\ 2 : v = 0\ and\ v^{'} > 0$$

$$Case\ 3 : v > 0\ and\ v^{'} = 0$$

$$Case\ 4 : v > 0\ and\ v^{'} > 0$$

Based on the expression of the link utilisation in Equation (4.7), an upper bound on the maximum volume of traffic that can be diverted at each iteration is investigated.

More specifically, by using the expression of $u_n(l)$, the expression of $u_{new}(l)$ can be re-arranged as follows:

$$u_{new}(l) = u_{cur}(l) + \frac{v - v^{'}}{c(l)} \tag{4.8}$$

By replacing the expression of $u_{new}(l)$ given by Equation (4.8) in the inequality (4.6), the following inequality can be obtained:

$$\forall l \in L \backslash S_{HU} \quad v - v^{'} < c(l) \cdot (u_n(l_{max}) - u_n(l)) \tag{4.9}$$

Given the expression of $u_{new}(l)$ in (4.8), it is known that $v \geq 0$ and $v^{'} \geq 0$, and as such, it can be stated that $v - v\prime \leq v$. It results that for any function $f(l)$, if $v \leq f(l)$ then $v - v\prime \leq f(l)$. It is then possible to obtain a constraint on the maximum volume $v$ that can be diverted over link $l$ as follows:

$$\forall l \in L \backslash S_{HU} \quad v < c(l) \cdot (u_n(l_{max}) - u_n(l)) \tag{4.10}$$

By definition, link $l$ is not included in the set $S_{HU}$. According to the definition of the set $S_{HU}$ this means that:

$$\forall l \in L \backslash S_{HU} \quad u_n(l) < (1 - \alpha) \cdot u_n(l_{max}) \tag{4.11}$$

By combining the inequalities in Equation (4.10) and Equation (4.11), an upper bound on the maximum volume $v$ that can be diverted towards link $l$ to satisfy Equation (4.6) can therefore be deduced as follows:

$$\forall l \in L \backslash S_{HU} \quad v < c(l) \cdot \alpha \cdot u_n(l_{max}) \tag{4.12}$$

As it can be seen, the so-derived upper bound depends on the actual capacity of the link $l$. In order to derive an upper bound independent of any link $l$, the expression of $c(l)$ needs to be replaced by a constant factor. Let $c_{bott}$ be the capacity of the bottleneck link, i.e. the link with the smallest capacity, among the links not included in the set $S_{HU}$. For any link $l$ the following inequality is thus satisfied:

$$\forall l \in L \backslash S_{HU} \quad c_{bott} \cdot \alpha \cdot u_n(l_{max}) \leq c(l) \cdot \alpha \cdot u_n(l_{max}) \tag{4.13}$$

Based on the latest and by using $u_n^{max}$ to represents the utilisation of link $l_{max}$, the upper bound $V_{max}$ on the maximum volume that can be diverted towards any link $l$ not including in the set $S_{HU}$ is given by:

$$V_{max} = c_{bott} \cdot \alpha \cdot u_n^{max} \tag{4.14}$$

The so-derived upper bound $V_{max}$ is determined by the bottleneck capacity in the set $\bar{S}_{l_{max}}^{S_i - D_j}$, by the current maximum utilisation in the network $u_n^{max}$, and by the parameter $\alpha$ of the set $S_{HU}$. In particular, it can be seen that the lowest is the value of the parameter $\alpha$, the lowest is the volume of traffic that can be diverted. The influence of the value $\alpha$ can be explained as follows. According to the definition of the set $S_{HU}$, the lowest is the value of $\alpha$, the closest to $u_{max}$ is the utilisation of the links in $S_{HU}$. Consequently, given that the gap between the utilisation of those links and $u_{max}$ is lower, there may be some links in $L \backslash S_{HU}$ with a utilisation closer to $u_{max}$. As such, the volume of traffic to divert towards these links needs to be lower not to overshoot the value of $u_{max}$ (Equation (4.6)).

**Diverted volume** The actual volume $v$ of diverted traffic for a selected flow $F(S_i - D_j)$ is defined as the current traffic volume $V((S_i - D_j), T)_n$ from that flow on one topology in $S_{l_{max}}^{S_i - D_j}$ divided by a factor $2^k$, where $k$ is an integer that varies between 1 and an upper bound $K$, i.e.

$$v = \frac{V((S_i - D_j), T)_n}{2^k} \quad k \in [1, K] \tag{4.15}$$

The value of $k$ is initially set to 1 and is iteratively incremented by 1 until the two following conditions are satisfied:

1. the total diverted traffic volume is less than the upper limit $V_{max}$, i.e.

$$v \leq V_{max}$$

2. the volume diverted from a topology $T$ in $S_{l_{max}}^{S_i - D_j}$ is not less than the actual volume of traffic currently routed in $T$, i.e.

$$v_T \leq V((S_i - D_j), T)_n$$

If it is not possible to find such a traffic volume $v$, the result of the algorithm is set to negative.

**Parameter K** To avoid diverting very little traffic at each iteration, the value of $K$ is bounded. However, the value of $K$ should not significantly penalise the outcome of the adaptation process, which is to re-balance the traffic load in the network. If the value of $K$ is too low, for instance, condition (1) defined above may permanently hold and as such, this may prevent the re-configuration algorithm from determining a traffic flow $F(S_i - D_j)$ to divert. In order to determine the order of magnitude of the value of $K$ (i.e. that does not penalise the output of the algorithm), simple approximations can be considered for the parameters $c_{bott}$, $\alpha$, $u_n^{max}$ and $V((S_i - D_j), T)_n$.

Given today's network capacity, let the capacity of the bottleneck link to be in the order of magnitude of few Mbps, i.e. $c_{bott} \sim 10^8$.
The value of the parameter $\alpha$ is typically in the order of 5 to 10%, and as such $\alpha \sim 10^{-1}$.
Finally, without loss of generality, let $u_n^{max} \sim 10^{-1}$.
Consequently, the value of $V_{max}$ is so that $V_{max} \sim 10^6$
Using the values provided by real traffic datasets [39][40][37], let $V((S_i - D_j), T)_n$ be in the order of magnitude of hundreds of Mbps [1], i.e. $V((S_i - D_j), T)_n \sim 10^8$. Given this approximation, let the volume of traffic to divert be written as a function $v(k)$ of the parameter $k$ so that:

$$v(k) = \frac{10^8}{2^k}$$

Figure. represents the profile of the function $v(k)$ for $k$ varying from 1 to 15.

As it can be observed in Figure (4.1), $v(k)$ is in the order of $10^6$ for $k$ equals to 6 and in the order of $10^5$ for $k$ equals to 10. The volume $v$ lower than $V_{max}$ for $k$ higher than 10. Consequently the value of $K$ is defined as being equals to 10.

### 4.5.3 Computing the adjustment factors

The volume of traffic shifted from $l_{max}$ is equally distributed across the topologies in $S_{l_{max}}^{S_i - D_j}$ and equally diverted towards the topologies in $\bar{S}_{l_{max}}^{S_i - D_j}$. Adjustment parameters

[1] Mbps stands for Megabit per second

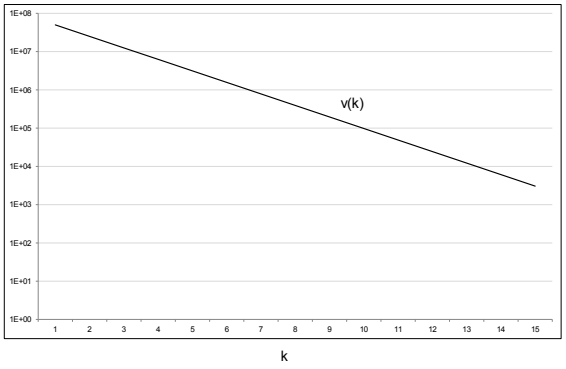


Figure 4.1: Function v(k)

$\delta^-$ and $\delta^+$ are therefore functions of the volume of traffic shifted away from $l_{max}$ and the number of topologies in each set, i.e.

$$\delta^- = \frac{v}{\left|S_{l_{max}}^{S_i - D_j}\right|} \tag{4.16}$$

$$\delta^+ = \frac{v}{\left|\bar{S}_{l_{max}}^{S_i - D_j}\right|} \tag{4.17}$$

# Chapter 5

# Signalling Communication Protocol

## 5.1 Introduction

In DACoRM, the traffic distribution in the network is controlled in an adaptive and decentralised fashion by the network source nodes equipped with traffic engineering adaptive capabilities that allow them to periodically re-configure the traffic splitting ratios of some traffic flows according to network current conditions. To support this adaptive re-configuration scheme, the source nodes are organised into an *in-network overlay* (INO), where the relevant entities can exchange information about the course of actions to follow in order to re-balance the network load.

More specifically, at each iteration of the adaptation process, one node among INO nodes is selected to be the Deciding Entity (DE), i.e. to initiate some re-configuration actions for the current re-configuration interval. As described in Section (4.2), the selection of the DE is the result of a local decision-making process performed independently by each source node and as such, does not require any interaction between the INO nodes. In the process of determining the re-configuration actions to apply, the DE may not always be able to compute new configurations for its local traffic flows. In this case, the re-configuration task needs to be delegated to other nodes in the INO, and as such, interaction between the different source nodes is required in order to exchange information.

While the previous chapter focuses on the details of the adaptation process, this chapter presents and describes a signalling communication protocol which facilitates the communication between the INO nodes and supports the delegation process. The actual communication model depends on the topology used to organise the nodes in the INO. In particular, the choice of the topology may directly influence the overhead (e.g. number of messages required) and convergence time incurred by the communication between the INO nodes. In this section two simple models to organise the nodes in the INO are presented. In the first model, nodes are connected according to a full-mesh topology where there exists a logical link between all the source nodes in the INO. In the second model source nodes are connected according to a ring topology, where each node is connected to only two other INO nodes. Based on the characteristics of these two models, a more complex topology relying a hierarchical structure where source nodes are connected in an hybrid fashion through a combination of the ring and the mesh approaches is also discussed.

In the rest of this chapter, it is considered that for reliability purposes, the proposed communication protocol relies on the Transmission Control Protocol (TCP) [49] as the underlying transport protocol.

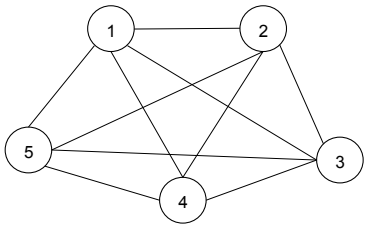


Figure 5.1: Source nodes organised according to a full-mesh model

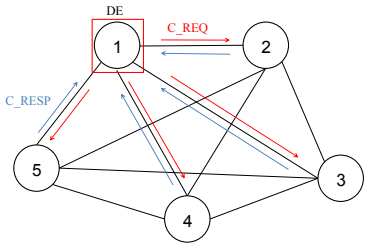


Figure 5.2: Communication model between the DE and the SEs in the full-mesh model

## 5.2 Full-mesh model

### 5.2.1 Structure of the in-network overlay

In the full-mesh model, INO nodes are connected in a full-mesh topology, as shown in Figure (5.1), where every node is logically connected to every other node.

The full-mesh model is characterised by a high degree of connectivity. Given $N$ nodes in the INO, the total number of logical links is equal to $\frac{N \cdot (N-1)}{2}$. Its structure is flat, i.e. all nodes are equal with regards of their functionality, and as such they can all be selected to be the DE. In addition, a greater flexibility in the choice of neighbours with which to communicate can be supported since all source nodes belong to the set of neighbours.

### 5.2.2 Communication model

In case the delegation process is triggered, communication between the DE and its neighbouring nodes (the SEs) in the INO is initiated by the DE in a star-fashion centered on the DE as depicted in Figure (5.2) Given the symmetry propriety of the structure, the communciation model is independent of the nodes selected to the DE and the SEs.

The communication protocol consists of three stages. Upon triggering a delegation process, the DE sends a delegation request - in the form of a $COMPUTE\_REQUEST$ ($C\_REQ$) message - to each of its neighbouring nodes (the SEs). The DE then enters a listening period where it waits for replies from all the SEs. Upon receiving a $C\_REQ$ message, the SEs execute the first phase of the re-configuration algorithm, as explained in Section (4.4), and copy the result into a $COMPUTE\_RESPONSE$ ($C\_RESP$) message that is sent back to the DE. In addition to compulsory information (such as the success status of any local re-configuration action), the $C\_RESP$ message can also include optional information that the DE can use when selecting a solution (see Section (4.3)). This information can be for instance the contribution in terms of volume of traffic of the local traffic flow for which ratio adjustments are proposed to the load of the link with the maximum utilisation in the network, $l_{max}$. Once the listening period expires, the DE considers all the different $C\_RESP$ messages and selects among the

| Field | Description |
|---|---|
| MESSAGE HEADER | |
| Length | Length of the packet |
| Action | COMPUTE / APPLY |
| Type | REQUEST / RESPONSE |
| Status | SUCCEED / FAIL |
| Fill bits | Unused bits |
| OPTIONAL INFORMATION ELEMENT | |
| ID | Type of appended information |
| Value | Value of the appended information |

Table 5.1: Structure of a message in the full-mesh model

successful configurations the one to apply. It then notifies the corresponding SE about its choice by sending a *APPLY_REQUEST* (*A_REQ*) message. Upon receiving this message, the chosen SE is responsible for enforcing the re-configuration it had proposed and for acknowledging the message by sending an *APPLY_RESPONSE* (*A_RESP*) back to the DE. Note that given that the communication model relies on TCP as the underlying transport protocol, the acknowledgement message is not compulsory.

In order for the DE to obtain the replies from all the nodes in the network, the lower bound of the waiting time at the DE needs to be higher than the maximum round-trip time (RTT) obtained between the DE and any of its neighbour nodes in the overlay. This is mainly driven by the physical distance between the source nodes [50][51].

### 5.2.3 Structure of signalling communication messages

TABLE (5.1) presents the structure of the messages used in the full-mesh model. Each message consists of a message header and can be extended with optional information elements (IE). IEs are typically appended by the SEs to *C_RESP* messages in order to provide the DE with additional information about the proposed local re-configurations. This information can be used by the DE to select a solution among the positive ones as explained in Section (4.3).

According to the action it supports, the message falls into two categories - *COMPUTE* (driving the execution of the re-configuration algorithm at SEs) or *APPLY* (driving the choice of the re-configuration decisions to enforce) - that is indicated in the field Action. The type of the message (*REQUEST* or *RESPONSE*) is indicated in the field Type. Both *COMPUTE* and *APPLY* messages can be of both types. It is important to note that *REQUEST* messages can only be sent by the DE as the initiator of the communication. The result of the re-configuration algorithm is indicated in the field Status; the default value is *FAIL* and is updated by the SEs upon sending the *C_RESP* message back to the DE. Each optional IE is uniquely defined by an identifier ID indicated in the field ID. The actual data related to the given identified element is copied in the field Value. One IE is typically appended to *C_RESP* messages by the SEs. In particular, this indicates the actual contribution in terms of volume of traffic of the local traffic flow for which ratio adjustments are proposed to the load of $l_{max}$.

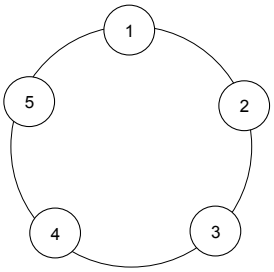


Figure 5.3: Source nodes organised in a ring

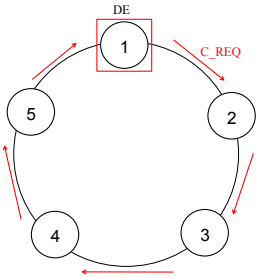


Figure 5.4: Communication model between the DE and the SEs in the full-mesh model

## 5.3 Ring model

### 5.3.1 Structure of the in-network overlay

In this model, INO nodes are connected according to a ring topology, as shown in Figure (5.3), where each node is connected to only two other nodes. Communication is unidirectional, which means that a node can only pass information to its immediate neighbour in the ring. To communicate with any other nodes, a message needs to be sent over the ring until it reaches its destination. The communication direction followed in the ring must be fixed but can be either anticlockwise or clockwise. Unlike the mesh model, the set of neighbours of any node is limited to its direct next hop node.

This model has a low degree of connectivity. The number of logical links is equal to the number of nodes in the INO. Its structure is flat. All nodes are equal and can assume the role of DE in an inter-exchangeable fashion.

### 5.3.2 Communication model

The delegation process in the ring model is supported by a two-stage communication protocol as follows. In a similar fashion to the full-mesh model, the communication model is independent of the node selected to be the DE. Upon triggering a delegation process, the DE sends a delegation request to only one of its neighbouring node as depicted in Figure (5.4).

As in the full-mesh model, the request comes in the form of a $C\_REQ$ message initiated by the DE. This then enters a listening period where it waits for the message to travel hop by hop through the ring until it reaches the DE again. Upon receiving the request message, the direct DE's neighbour determines the result of the re-configuration algorithm executed locally. If the result is negative, the message is directly passed to the next neighbour node. If the result is positive, the node reports the local solution in the message, especially it indicates the contribution in terms of volume of traffic of

| Field | Description |
|---|---|
| MESSAGE HEADER | |
| Length | Length of the packet |
| Action | COMPUTE / APPLY |
| Type | REQUEST / RESPONSE |
| Status | SUCCEED / FAIL |
| Fill bits | Unused bits |
| OPTIONAL INFORMATION ELEMENT | |
| ID | Type of appended information |
| Value | Value of the appended information |

Table 5.2: Structure of a message in the ring model

the corresponding local flow to the load of $l_{max}$. It also indicates its identity (e.g. its IP address) for further identification of the relevant SE and sends the message to its neighbour node. Upon receiving the request message, the next hop node analyses the content of the message to decide whether or not to replace the current re-configuration result with its own result. This is if the contribution is higher than the one related to the re-configuration currently reported. In that case, the node replaces the current information with the new one (i.e. updates the contribution and node's identity) and forwards the message to the next hop node. Once the message reaches the DE it is analysed, and, if a successful re-configuration is reported, the DE sends a *A_REQ* message to the address of the corresponding SE. While this message can be propagated through the ring, it can also be sent directly to the SE in the same manner as in some peer-to-peer file sharing systems where a direct connection is established between peers once the content has been located (for instance in the peer-to-peer file-sharing system Gnutella [52]). Upon receiving the *A_REQ* message, the SE is responsible for enforcing the re-configuration it had proposed. The chosen SE may acknowledge the message by sending an *APPLY_RESPONSE* (*A_RESP*) back to the DE.

Compared to the full-mesh model, where the final selection of a re-configuration action is left to the DE, each node in this model is responsible for determining whether the local solution is more appropriate than the one currently reported. The DE is not responsible for applying any selection rule.

### 5.3.3 Structure of signalling communication messages

The structure of the messages used in the ring model is similar to the one used in the full-mesh model (see TABLE (5.2)). Each message consists of a message header and can be extended with optional information elements (IE).

Messages fall in two categories according to the action it supports - *COMPUTE* (driving the execution of the re-configuration algorithm at SEs) or *APPLY* (driving the choice of the re-configuration decisions to enforce) - that is indicated in the field *Action*. The type of the message (*REQUEST* or *RESPONSE*) is indicated in the field *Type*. Unlike this approach, the type *RESPONSE* is only defined for *APPLY* messages. The result of the re-configuration algorithm is indicated in the field *Status*; its default value is *FAIL* and is updated by the first node in the ring that obtain a successful result locally. Each optional IE is uniquely defined by an identifier ID indicated in the field ID and the actual data related to the given identified element is copied in the field *Value*. Two IEs are typically appended to *C_REQ* messages in the ring model. The first IE relates to the identity of the INO node that reports the positive solution. The second

relates to the associated actual value of the contribution in terms of volume of traffic of the local flow for which ratio adjustments are proposed to the load of $l_{max}$. In case an intermediate node decides to replace to current solution, it needs to update the value of the two IEs.

### 5.3.4 Full-mesh model versus ring model

In order to minimise the overhead incurred by the delegation process, the size of coordination messages needs to be small. The messages are designed such as the size is typically less than 10 bytes. However the actual number of messages exchanged depends on the model. Assuming an INO of $N$ source nodes, the total number of coordination messages exchanged during the delegation process in the full-mesh approach is the sum of $(N-1)$ $C\_REQ$, $(N-1)$ $C\_RESP$, 1 $A\_REQ$ and 1 $A\_RESP$, i.e. a total of $2N$ messages. In the ring model, the number of messages is the sum of 1 $C\_REQ$, 1 $A\_REQ$ and 1 $A\_RESP$, i.e. a total of 3 messages. Although the number of messages is independent of the number of INO nodes in the ring model, it linearly increases with the number of INO nodes in the mesh approach. As such, the ring model offers a better performance than the full-mesh model in terms of communication overhead.

However the performance of the communication model is also driven by the total time required to complete the delegation process, i.e. for the DE to obtain and select a new configuration. It can be inferred that the waiting time for the DE to obtain the best re-configuration in the ring is relatively long, as the message needs to traverse all the nodes attached to the INO. In addition, due to the nature of the model, the actual waiting time increases with the number of nodes. In the full-mesh model, the waiting time is mainly driven by the size of the network in terms of distance rather than by the number of nodes. For the delay not to be an issue in practice, the time required to perform re-configurations needs to be kept small (maximum few seconds) compared to the frequency at which adaptation is invoked (order of tens of minutes).

In addition to these differences in terms of performance, the topological characteristics of the two models also offer different advantages. In particular, given its high degree of connectivity, the full-mesh topology is more robust than the ring topology. In addition, it can provide a greater flexibility. Since all nodes are connected to each others, it is possible for the DE to select the number of nodes a message can be sent directly to. Also, given that the DE has a global knowledge of all the possible adaptations proposed by the other nodes, it can always select the b̈estsolution according to the selection policy applied. In the ring model, the selection is performed locally by each node along the ring according to the information received from its neighbour node only. Indeed for the DE to obtain a global knowledge of all the possible adaptations proposed by the other nodes, it would be necessary that each node along the ring appends its local solution to the request message. Although this may not be an issue if the number of nodes in the ring is not significant, this solution does not scale well since the size of the message would increase proportionally with the number of nodes.

The characteristics of the two models are summarised in TABLE (5.3).

The performance of the two approaches with respect to the communication overhead and the total re-configuration delay is evaluated in the next chapter (Section 6.2).

## 5.4 Towards an hybrid model?

### 5.4.1 Structure of the in-network overlay

In order to effectively demonstrate the actual coordination process between the nodes in the INO, the previous section have been focussing on two simple and lightweight

| | **Full-mesh model** | **Ring model** |
|---|---|---|
| **Number of messages** | 2N | 3 |
| **Delegation delay** | Driven by the size | Driven by the number of nodes |
| **Neighbour selection flexibility** | High | Restricted |
| **Selection process** | Centralised on the DE | Local hop-by-hop |

Table 5.3: Full-mesh model vs. ring model

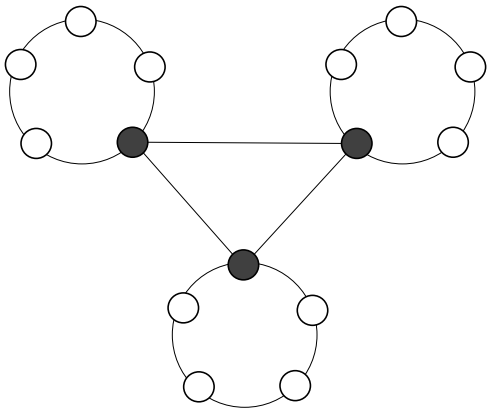

Figure 5.5: Source nodes organised in an hybrid model

(in terms of implementation and maintenance cost) overlay topologies. However due to their characteristics, both models present some limitations as the number of nodes in the INO increase. In particular, this involves an increase in the number of messages in the full-mesh approach, while this significantly affects the total delay to obtain a solution in the ring model.

In this section, a more advanced organisational model that can overcome the limitations of the two previous approaches in terms of communication overhead and delay is investigated. This model relies on an hybrid topology that consists in a combination of the ring and the full-mesh approaches. Compare to the previous models that rely on a flat topological structure, the hybrid model uses a hierarchical structure to connect the nodes.

More specifically INO nodes are partitioned into several subsets of equivalent size, so that the nodes in each subset are connected according to a ring topology. The different rings are then connected to each other according to a full-mesh topology, so that it is possible for nodes located in different rings to communicate between each others. The structure of the hybrid model is depicted in Figure (5.5).

Rings are connected to each other through intermediate nodes, so that there is only one intermediate node in each ring, as depicted in Figure (5.5), where the three local rings are connected to each other through the dark nodes. Unlike the previous approaches, the delegation process in the hybrid model consists in two phases. Upon triggering delegation, the Deciding Entity initiates the search inside its local ring according to the communication model used in the ring model. In case no positive solution can be found in the local ring, the Deciding Entity extends the search to the other nodes in the INO by sending a request to each of the other rings, so that the requests

are forwarded in a full-mesh fashion to the other rings through the intermediate node, i.e. the node directly connected to the distant rings. Upon receiving such a request, the intermediate nodes in each distant ring are responsible for forwarding the message in their local ring according to the ring communication model and for returning it back to the intermediate node in the original ring.

Since by design any node in the INO can be selected to be the Deciding Entity, there may be cases where an additional message is required in the communication protocol. This is especially the case when the node selected to be the Deciding Entity is not an intermediate node. In such a case, upon triggering an extended search, the Deciding Entity first sends an *extended search request* notification directly to the intermediate node in its local ring. Upon receiving the notification, the later is then responsible for forwarding an *extended search request* message to the other intermediate nodes in the INO. Compared to the full-mesh and ring models, a level of hierarchy is therefore introduced between the nodes.

### 5.4.2 Design challenges

Although a fairly simple example of an hybrid topology is depicted in Fig. X, the design of such a model raises several challenges in practice. The topology needs to be so that the advantage of the ring model in terms of limited number of communication messages needed is retained, while the total communication delay incurred does not become critical. This subsection does not aim at presenting a detail solution to design the hybrid topology but instead, discusses some of the main research challenges raised by this approach. An extended analysis of this model will be part of the future directions that will be tackled by this Ph.D. work.

As for the number of communication messages required, two cases can be distinguished depending whether an extending search is required or not. In the case where a local search is enough for the Deciding Entity to obtain a positive solution, only messages required in the ring model are required, i.e. a total number of three messages. In the other case, messages need to be sent to each ring in the INO, so that the total number of messages is directly driven by the number of rings.

More precisely, let $x$ be the number of nodes in each local ring and $N$ the number of INO nodes. As such, the number of local rings is equal to $\frac{N}{x}$. The total number of messages is then driven by the sum of $\frac{N}{x}$ extended search requests and $\frac{N}{x}$ extended search replies, i.e. $\frac{2 \cdot N}{x}$ messages. Even in the worst case where each ring has a size of two (i.e. there are only two nodes in each ring), the total number of messages is equal to $N$, that is half as the number of messages required in the full-mesh approach.

The main design challenge is to determine the optimal size of each ring in order to minimise the total communication delay incurred. In particular, the expected delay needs to be so that it does not exceed the total delay incurred if a ring model was used instead. At the local ring level, this involves that the number of nodes in each ring is set so that the communication delay incurred does not exceed the communication delay expected if these nodes were connected in a full-mesh fashion. It is therefore necessary to determine the maximum size of each ring.

In addition, another main issue to consider is the choice of the metrics to use in order to cluster the nodes in the different local rings. The choice of a metric is critical for the performance of the clustering algorithm [Book clustering]. Given the communication delay requirements, the proximity in distance between the nodes in the physical network is surely an important metric to consider since it may affect the total delay in the ring. However, other more advanced parameters may also be investigated. In order to optimise the probability of finding a solution in the local ring, it may be interesting, for instance, to capture the likelihood for a node to positively reply to a delegation request

received from another INO node, so that nodes with highest likelihood may be clustered together.

Another research question to further investigate relates to the choice of intermediate nodes in each local ring. Selecting only one intermediate node in each topology facilitates the management of the topology but introduces a degree of hierarchy between the nodes. In addition, it introduces a degree of complexity to the communication protocol since messages from the Deciding Entity need to traverse the intermediate node of its local ring to reach the other subsets of node in the network.

# Chapter 6

# Evaluation

In order to determine the overall efficiency of the proposed scheme, the different mechanisms used in DACoRM need to be evaluated. This chapter presents the results of the evaluation of DACoRM adaptive scheme (i.e. the adaptation algorithm) and of the communication protocol. More specifically, the first section investigates the performance of the adaptive scheme, especially the gain that can be achieved in terms of resource utilisation and analyses the influence of different parameters. The second section focuses on the evaluation of the communication protocol described in Section (5).

## 6.1 Performance of the adaptive scheme

This section focuses on evaluating the performance of the adaptive resource management scheme, i.e. to evaluate the performance that can be achieved in terms of resource utilisation through the adaptation procedure used in DACoRM to re-adjust traffic splitting ratios. The influence of the different parameters of the adaptation algorithm on the overall performance of the adaptive scheme is also investigated.

### 6.1.1 Experiments setup

The evaluations are performed through a self-implemented JAVA software. In a similar fashion to the TOTEM traffic engineering tool [53], the software permits to obtain relevant network statistics (i.e. link utilisation) given a network topology, specific traffic engineering settings (e.g. traffic splitting ratios) and traffic matrices. The traffic splitting ratios for a specific traffic demand (i.e. traffic matrix) are computed through the implementation of the adaptation procedure used in DACoRM. More specifically, the main programme rely on three main functionalities defined as follows:

- the monitoring functionality that is responsible for computing network statistics
- the adaptation functionality that is responsible for executing the adaptation process
- the re-configuration functionality that is responsible for determining and computing new splitting ratios according to the re-configuration algorithm

These functionalities rely themselves on some common inputs from the main programme, i.e.

- a physical network topology for which a set of source nodes is defined
- a set of pre-computed virtual topologies from which the set of paths between each source-destination pair in the physical network is extracted

- a traffic matrix related to the incoming demand at the source nodes

Given a new traffic matrix, the main programme invokes the adaptation functionality that defines the value of the parameter $\alpha$ and the maximum number of permitted iterations. This in turn invokes the monitoring functionality that is responsible for computing network statistics given the input traffic matrix and the currently enforced traffic splitting ratios. These statistics are then used to extract specific information, especially the link with the maximum utilisation in the network, the set of highly utilised links $S_{HU}$ and the identity of the relevant Deciding Entity. Based on the information obtained through the monitoring functionality, the adaptation functionality then calls the re-configuration functionality that is responsible for executing the re-configuration algorithm according to the attributes of the source node selected as the Deciding Entity.

Since the actual communication between the different source nodes in the INO is not implemented, the programme relies on a functionality that emulates the delegation process in case the delegation condition is triggered by the re-configuration functionality. More precisely, the delegation process is emulated by applying the method that implements the first phase of the re-configuration algorithm to all the source nodes other than the Deciding Entity, and by returning their results to the Deciding Entity node for further operations.

The integrity of the different functionalities of the Java software is verified through both dynamic (experimentation) and static (analysis) verifications. To ensure that the software integrity is satisfied, all classes have been tested through functional test and the code are verified based on the code conventions. The experiments are run on a machine with a 2.53 GHz Intel Core 2 Duo processor and 4 GB memory.

### 6.1.2 Topologies and traffic datasets

In order to obtain realistic results, the performance of the adaptive scheme is evaluated in two real Point of Presence (PoP) level topologies, namely the Abilene network [54] and the GEANT network [55], for which real traffic measurement datasets, as well as link capacities and link weights, are available.

The Abilene network topology [54] consists of 12 PoP (or nodes) and 15 bi-directional links. The links have a capacity of 9.2Gbps, except the link (ATLA - IND) that has a capacity of 2.8Gbps. Traffic traces for the Abilene network are available at 5 minutes intervals [54].

The GEANT network topology [55] consists of 23 PoP and 38 bi-directional links. Compare to the Abilene network, link capacity is more diverse in the GEANT network. This is distributed as follows: 16 links have a capacity equal to 10Gbps, 16 links have a capacity equals to 2.4Mbps, 2 links have a capacity of 622Mbps and 4 links have a capacity 155 Mbps. In addition, traffic traces are available at a larger timescales, at 15 minutes intervals [45].

In order to represent a wide range of traffic conditions, traffic matrices over a period of 7 days are considered in the evaluation. Note that for consistency between the two topologies, adaptation is performed at a frequency of 15 minutes in both topologies.

### 6.1.3 Computing the virtual topologies

The virtual topologies are computed according to the algorithm presented in Chapter (3), Section (3.3). The characteristics of the two networks are summarised in TABLE (6.1). The table indicates for each network topology if there exists constrained links, i.e. links that cannot be removed whatever is the number of virtual topology considered[1],

[1]These are the links that have a connectivity equal to one as explained in Section (3.3)

| | Nb. of nodes | Nb. of bi-directional links | Constrained link | Nb. of constrained links |
|---|---|---|---|---|
| **GEANT** | 23 | 37 | NO | 0 |
| **Abilene** | 12 | 15 | YES | 1 |

Table 6.1: Topology characteristics

| | GEANT | Abilene |
|---|---|---|
| **2 topologies** | 8.11% | 28.57% |
| **3 topologies** | 83.78% | 78.57% |
| **4 topologies** | 91.89% | 100% |
| **5 topologies** | 100% | - |

Table 6.2: Percentage of satisfying links according to the number of topologies

and if so, their number.

TABLE (6.2) indicates the percentage of links that satisfy the path diversity requirements in each network according to the number of virtual topologies computed.

The optimum number of virtual topologies represents the number of virtual topologies required in order to satisfy the requirements for all of the links in a specific network. The optimum number is equal to five topologies in the GEANT topology, whereas it is equal to four topologies in the Abilene topology. These results are consistent with [41] that argues that a small number of topologies (typically a maximum of 6 topologies) are required in order to satisfy all the links. In the rest of this chapter, the optimum number of virtual topologies for a specific network is referred as the OPT-VT.

### 6.1.4 Performance in terms of gain

The objective of the experiments is to quantify the gain that can be achieved by the adaptive scheme in terms of resource utilisation. In order to evaluate the scheme over a wide range of traffic conditions, traffic matrices over period of one week at 15 minutes interval are considered, and as such this represents 672 traffic matrices. In order to quantify the actual gain, three different schemes are considered:

- **Original scheme**: the original link weight settings are used in the original topology and no adaptation is performed
- **DACoRM scheme**: virtual topologies are used to provide path diversity and periodic adaptation of the splitting ratios is performed. Initially the traffic between any source-destination pair is split evenly across the different available paths.
- **The optimum**: the TOTEM toolbox is used to compute the optimal maximum utilisation for each traffic matrix. TOTEM defines the routing problem as a Multi Commodity Flow problem and uses a linear programming formulation to solve it [56].

For each scheme, the maximum utilisation in the network (max-u) obtained for each measurement period (i.e. for each traffic matrix $TM_i$, $\in i\,[0;672]$) is determined. The performance of the Original scheme and DACoRM scheme are then compared against the optimum. The incentives for this methodology rely on the fact that existing online traffic engineering approaches can achieve close to optimum performance only (e.g.

| | GEANT | Abilene |
|---|---|---|
| **Number of topologies** | 5 | 4 |
| $\alpha$ (%) | 5 | |
| **Max number of iterations** | 50 | |

Table 6.3: Experiment parameters

[28][26][30]). As such, instead of choosing an existing algorithm to compare against, directly comparing to the optimum can provide the most relevant evaluation factor.

The max-u obtained for a traffic matrix $TM_i, \in i\,[0;672]$ is noted $u^{max}_{TM_i}$. It is further noted $u^{max,opt}_{TM_i}$ if it represents the result obtained with the optimum scheme.

To compare the performance of the Original scheme and DACoRM scheme, three evaluation factors are considered:

- **Average deviation (AD)**: this indicates the average deviation of the max-u from the optimum over the set of measurements (i.e. across all the traffic matrices $TM_i, \in i\,[0;672]$). The factor AD is computed as follows:

$$AD = \frac{\sum_{i=0}^{672} \frac{u^{max}_{TM_i}}{u^{max,opt}_{TM_i}}}{672}$$

- **Percentage Near-Optimal Max-U (PNOM)**: this indicates the percentage of adaptation sequences (i.e. one adaptation sequence per traffic matrix) for which near-optimal performance can be achieved. Near-optimal performance is defined to be such as the obtained max-u is within 10% of the optimum. Let $\pi_i$ be a parameter defined so that:

$$\pi_i = \begin{cases} 1 & u^{max}_{TM_i} \leq 1.1 u^{max,opt}_{TM_i} \\ 0 & otherwise \end{cases}$$

  The factor PNOM is then defined as follows:

$$PNOM = \frac{\sum_{i=0}^{672} \pi_i}{672} \cdot 100$$

- **Highest Maximum Utilisation (HMU)**: this indicates the highest max-u obtained in the network across all the traffic traces during the one week period.

$$HMU = max_{TM_i}\left(u^{max}_{TM_i}\right)$$

The settings of the different parameters used in the DACoRM scheme to perform the experiments are summarised in TABLE (6.3). The number of topologies considered is equal to the optimal value OPT-VT in each network. The value of $\alpha$ is set to 5% and the maximum number of permitted iterations is equal to 50. Note that these settings are referred as the standard settings and unless explicitly stated, these are the settings used to perform the experiments.

The results of the experiments for the different evaluation factors are presented in TABLE (6.4)[2].

The results show that near-optimal performance can be achieved by DACoRM in both the GEANT and the Abilene networks, with an average deviation of less than 10%

[2]All the results are in percentages

| | GEANT (%) | | | Abilene (%) | | |
|---|---|---|---|---|---|---|
| | **AD** | **PNOM** | **HMU** | **AD** | **PNOM** | **HMU** |
| **Optimum** | - | - | 43.79 | - | - | 12.19 |
| **Original** | 89.88 | 0 | 96.91 | 54.34 | 0 | 21.16 |
| **DACoRM** | 5.49 | 96.14 | 45.44 | 7.53 | 94.93 | 13.1 |

Table 6.4: Experimental results for the different evaluation factors

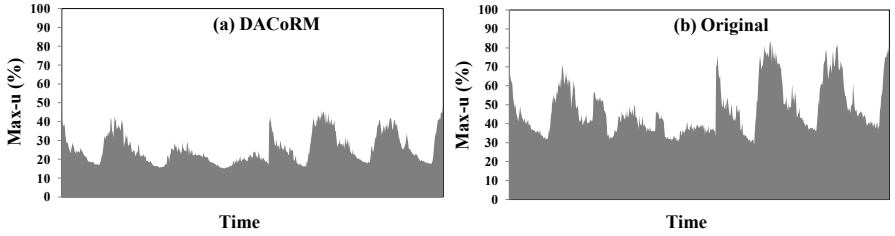


Figure 6.1: Evolution of max-u at 15 minute intervals using (a) DACoRM, and (b) Original scheme, for the GEANT network

from the optimal, respectively around 5.5% and %. The PNOM factor indicates that the near-optimal performance are achieved for respectively 96% and % of the traffic matrices considered, which shows that the proposed scheme performs uniformly well. In addition the HMU factor shows that the highest network utilisation observed in the DACoRM scheme is less than 1% (i.e. $\frac{45.44}{43.79}$) of the highest optimal utilisation in the network. Unlike DACoRM, poor performance is achieved with the Optimal scheme in both the GEANT and Abilene networks. The average deviation is around 54% higher than the optimum in the Abilene network and almost 90% higher in the GEANT network. In addition near-optimal performance are never achieved.

To observe the dynamics of the traffic traces used for the experiments in the GEANT and in the Abilene networks, the evolution of max-u at 15 minute intervals for a) DACoRM and b) the Original scheme is presented in Figure (6.1) and Figure (6.2), respectively. As it can be seen, DACoRM can achieve a significant gain in terms of resource utilisation in the GEANT network. The max-u obtained in DACoRM scheme is permanently much lower than the max-u obtained in the Original scheme. Similar results are shown in the Abilene network. Note that the Abilene network shows smoother traffic dynamics than the GEANT network.

These results show that DACoRM outperforms the Original scheme, with a gain of more than 100% in terms of resource utilisation. DACoRM can substantially reduce the utilisation of network resources.

### 6.1.5 Influence of the different parameters

In this section, the influence of different parameter settings on the performance of the adaptive algorithm is investigated. Three parameters of the algorithm are considered as follows:

- **Number of topologies (NbT)**: the number of virtual topologies used to provide path diversity. All links satisfy the path diversity requirements with OPT-T number of virtual topologies in each network.

- **Parameter** $\alpha$: this parameter indicates which links are to be considered as highly

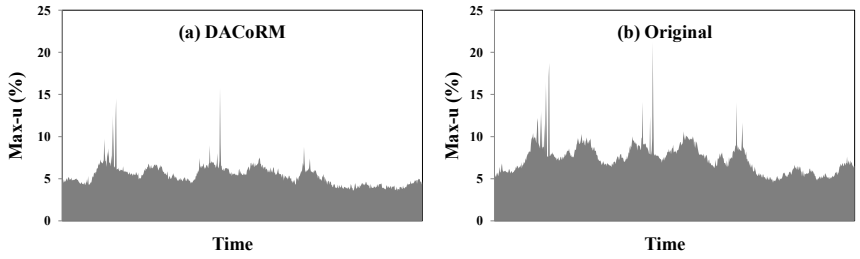


Figure 6.2: Evolution of max-u at 15 minute intervals using (a) DACoRM, and (b) Original scheme, for the Abilene network

| | GEANT (%) | | | Abilene (%) | | |
|---|---|---|---|---|---|---|
| | **AD** | **PNOM** | **HMU** | **AD** | **PNOM** | **HMU** |
| **Optimum** | - | - | 43.79 | - | - | 12.19 |
| **Original** | 89.88 | 0 | x | 54.34 | 0 | 21.16 |
| **NbT = 2** | 90.89 | 0 | 86.03 | 78.57 | 0 | 23.15 |
| **NbT = 3** | 15.10 | 47.77 | 57.04 | 13.4 | 29.50 | 15.11 |
| **NbT = 4** | 4.45 | 85.30 | 50.68 | 7.53 | 94.93 | 13.1 |
| **NbT = 5** | 5.49 | 96.14 | 45.44 | - | - | - |

Table 6.5: Influence of the number of virtual topologies

utilised (compared to the actual maximum utilisation in the network). As explained in Section (4.1), the value of $\alpha$ defines the gap between the links that can receive diverted traffic and links that should avoid receiving traffic, and as such, it directly affects the maximum volume of traffic that can be diverted at each iteration.

- **Max number of iterations (NbIt)**: this parameter indicates the maximum number of permitted iterations of the adaptation process and therefore may influence the convergence speed of the algorithm.

**Influence of the number of topologies** In order to evaluate the influence of the number of virtual topologies on the performance of DACoRM, experiments are run with the same measurement dataset but with different numbers of virtual topologies used to provide path diversity. The two other parameters are fixed as follows: $\alpha = 5$ and NbIt = 50.

The results obtained for the three evaluation factors according to the number of virtual topologies used are presented in TABLE (6.5). Although the results obtained with the Original scheme and the Optimum are not affected by the number of virtual topologies, these results are also reported in the table to provide a baseline for comparison.

As expected, the best performance is obtained in both networks when the optimal number of topologies is used to provide path diversity (i.e. all links satisfy the path diversity requirements). In addition, the adaptive scheme can achieve better performance as the number of virtual topologies increased. Performance obtained with two topologies do not offer any gain compared to the Original scheme neither in the GEANT network and the Abilene network. Although the PNOM factor is only around 48% in the GEANT network in the scenario where three topologies are used, DACoRM signifi-

| | GEANT (%) | | | Abilene (%) | | |
|---|---|---|---|---|---|---|
| | **AD** | **PNOM** | **HMU** | **AD** | **PNOM** | **HMU** |
| $\alpha = 2$ | 6.89 | 87.04 | 56.75 | 13.23 | 65.88 | 13.8 |
| $\alpha = 5$ | 5.49 | 96.14 | 45.44 | 7.53 | 94.93 | 13.1 |
| $\alpha = 10$ | 5.68 | 96.00 | 45.94 | 8.57 | 93.6 | 13.45 |
| $\alpha = 15$ | 6.78 | 92.33 | 46.14 | 10.47 | 80.17 | 13.74 |

Table 6.6: Influence of the value of the paramater $\alpha$

cantly performs better than the Original scheme and can achieve substantial gain in the network in terms of resource utilisation, with an average deviation of around 15% from the optimum. Same results can be observed in the Abilene network, where the average deviation from the optimum is around 13

While the average deviation obtained in the scenario with four topologies (GEANT) is slightly lower than the one obtained in the OPT-VT scenario, the overall performance is not as good as the performances achieved in the latter one since the PNOM is equal to 85% only and the HMU parameter is higher. The reason for the scheme not to perform as uniformly well as in the scenario with the optimal number of topologies is the following. In this case, given some links do not satisfy the path diversity requirements, diverting traffic from any link in the network is not possible. As such, the algorithm has less flexibility to move traffic flows towards different links, which is reflected by a lowest value of PNOM.

It is worth noting that these results show that near-optimal performance can be achieved by using only a small number of routing topologies, keeping the size of routing tables a small multiple of today's ones. This is encouraging regarding the scalability and applicability of the approach.

**Influence of the parameter $\alpha$** In order to evaluate the influence of the parameter $\alpha$ on the performance of DACoRM, experiments are also run with different value of $\alpha$. The other parameters are fixed. For each network, the number of virtual topologies is equal to OPT-VT and the maximum number of iterations is equal to 50. Four different values for $\alpha$ are considered, $\alpha$ equals to 2, 5, 10 and 15%. The results obtained for the three evaluation factors for different values of $\alpha$ are presented in TABLE (6.6).

It can be seen that the performance of the DACoRM scheme is not dramatically affected by the value of the parameter $\alpha$ in the GEANT network. While in both networks, the performance achieved with $\alpha = 5$ and $\alpha = 10$ is similar, the performance is slightly deteriorated when $\alpha = 2$ as the value of the PNOM factor is lower. This can be explained as follows. A lower value of the parameter $\alpha$ indicates a lower gap between the links that can receive diverted traffic and links that should avoid receiving traffic, and is reflected by a lower maximum volume of traffic that can be diverted at each iteration. This results in adding some constraints to the ratio adjustment steps and as such, to the choice of a feasible traffic flow to modify. Given these constraints, re-configuration can be performed for only a limited number of traffic flows, which is not be enough to re-balance the traffic load. In addition it can be observed that the performance obtained with $\alpha = 15$ is also slightly inferior than the ones obtained with $\alpha = 5$ or $\alpha = 10$. In this case, although larger ratio adjustment steps are permitted, the number of potentially critical links may increase, which may affect the choice of alternative solutions since the algorithm avoid diverting traffic towards those highly utilised links. These two phenomena are accentuated in the Abilene network where the number of completely disjoint paths is smaller than in the GEANT network given the

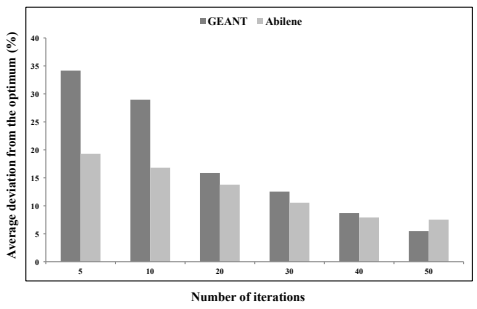


Figure 6.3: Evolution of the average deviation from the optimum according to the maximum number of permitted iterations in the GEANT network and the Abilene network

physical characteristics of the network.

**Influence of the maximum number of permitted iterations** The influence of the maximum number of permitted iterations on the performance of DACoRM is also evaluated. Experiments are run with different values for the maximum number of permitted iterations. In both network, the optimal number of virtual topologies is used and the parameter $\alpha$ is equal to 5%.

The evolution of the average deviation of the maximum utilisation from the optimum according to the maximum number of permitted iterations NbIT in the GEANT network and the Abilene network is presented in Figure (6.3).

As it can be observed, the average deviation of the maximum utilisation from the optimum decreases as the maximum number of iterations increased. In fact the convergence of the algorithm can be affected by the initial settings of the splitting ratios at the beginning of each adaptation interval (i.e. every 15 minutes). More specifically, if no significant change occurs in the traffic demands between two consecutive monitoring intervals, previously computed splitting ratios are already set so that traffic is already well balanced in the network. As such, no major modification of the settings is required, which may be obtained in a small number of iterations. However, in case of a dramatic change in the traffic demands between two consecutive monitoring intervals, significant modifications of previously computed splitting ratios may be required in order to re-balance the traffic load.

In the experiments, splitting ratios are initialised to random values (satisfying traffic splitting ratios constraints, Equations (3.1) and (3.2)) and then modified according to the adaptation algorithm.

### 6.1.6 Computing overhead

This section focuses on the time required to execute the adaptation process. More specifically, the time required for a source node to determine new splitting ratios is investigated. The analysis of the time-complexity of the re-configuration algorithm is presented in Section (4.4.4) of this thesis. It has been shown that the complexity of the first phase of the algorithm is driven by the number of locally originating traffic flows to analyse. Here the objective of the experiments is to compare the order of magnitude of the actual computing time for a source node to determine new splitting ratios in the GEANT and in the Abilene network. It is worth mentioning that only an order of magnitude is investigated and not the actual value. The measure of the actual execution time of the algorithm may depend on various factors, especially on the code

| | GEANT | Abilene |
|---|---|---|
| **Average execution time (ms)** | 4.13 | 1.46 |

Table 6.7: Average execution time

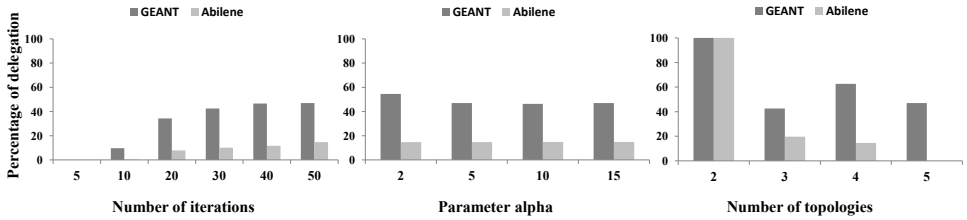


Figure 6.4: Evolution of the frequency of delegation according to (a) the maximum number of permitted iterations, (b) the value of the parameter $\alpha$, and (c) the number of virtual topologies in the GEANT network and the Abilene network

implementation and the platform used to execute the JAVA programme. As such, the results obtained need to be considered as relative to the execution environment and to be compared among themselves only. The average execution time is presented in TABLE (6.7). It is determined over 10,000 samples.

On average it takes less than few milliseconds for DACoRM to execute the first phase of the re-configuration algorithm. The average execution time is higher in the GEANT network than in the Abilene network. This is consistent with the fact that the number of local traffic flows in the GEANT network is larger than in the Abilene network.

### 6.1.7 Delegation analysis

In order to limit the overhead incurred by the delegation process, the adaptation procedure is designed such as splitting ratio adjustments are performed locally by the Deciding Entity as much as possible. Different factors may affect the frequency at which delegation is invoked by a source node. In order to analyse how often the delegation process is triggered, each time the re-configuration algorithm is executed by the Deciding Entity source node, it is monitored if delegation is triggered. Delegation is triggered if the conditions of delegation are satisfied.

The frequency of delegation is then defined as the ratio of the number of times delegation is triggered to the number of times the re-configuration algorithm is executed at the Deciding Entity source node during the adaptation process. The result is then average over all the set of experiments (i.e. over the one-week period). A low value indicates that delegation is triggered less often and as such that the scheme performs well with respects to the overhead incurred by delegation, whereas a large value indicates poorer performance. Figure (6.4) represents the evolution of the frequency of delegation according to (a) the maximum number of permitted iterations, (b) the value of the parameter $\alpha$, and (c) the number of virtual topologies in the GEANT network and the Abilene network .

It can observed that the frequency at which delegation is invoked increases with the number of iteration. Indeed as the number of iteration increases, the traffic load in the network is more evenly balanced so that the number of links with a utilisation close to the maximum utilisation increases. As such, the number of links in the set $S_{HU}$ becomes larger, which limits the choice of the links towards which traffic can be

diverted. As a result, the probability for the Deciding Entity node to find a positive solution locally is decreased, and delegation is more likely to be triggered.

The value of the parameter $\alpha$ does not significantly affect the frequency at which delegation is triggered, which is permanently around 50% of the time in the GEANT network and only around 15% in the Abilene network. The frequency of delegation is also affected by the number of virtual topologies used. The worst case is obtained when the number of topologies is equal to two, where delegation is triggered at each iteration. In this case, very few links can satisfy the path diversity requirements, which means that traffic can be diverted from only a small subset of links. The probability for a Deciding Entity source node to determine locally a successful re-configuration is therefore lower, which results in delegation. In the other cases, the number of topologies does not significantly affect the frequency of delegation.

As it can be observed from these results, delegation is triggered by the Deciding Entity in almost half of the cases in the GEANT network, whereas it is triggered around 15% of the cases in the Abilene network. Although the parameters of the algorithm may influence the frequency of delegation, this is, in fact, also affected by the procedure used to select the Deciding Entity. As explained in Section (4.2), this selection procedure relies on a static rule that does not evaluate whether a node can effectively perform or not some re-configurations given the current network and traffic conditions. As such, even if a node is unable to perform any local adjustment, it can still be selected to be the Deciding Entity. In particular, according to the selection rule currently used, a source node that is so that all its incoming traffic demands are null can be selected to be the Deciding Entity, despite the fact that in this case, it will not be able to perform any adjustment. Although such a situation is unlikely to occur in practise, it simply illustrates the limitations of the current selection rule.

A solution where only those of the source nodes that can effectively perform some re-configurations given the current network and traffic conditions are considered to be selected to be the Deciding Entity may therefore reduce the frequency at which delegation is invoked.

## 6.2 Evaluation of the communication protocol

In addition to the performance in terms of resource utilisation gain, the overall performance of DACoRM also relies on the convergence time and cost (in terms of management overhead) of the scheme. Different factors may influence the time required to complete the adaptation process, such as the physical characteristics of the network and the execution of the delegation process at different iterations of the adaptation cycle. The actual time to execute one iteration depends on the execution time of the re-configuration algorithm described in Section (5). In particular, in the best case where no delegation is required, the execution time of the algorithm is given by its first phase. In this case, it takes on average few milliseconds only for a source node to determine new splitting ratios for the network topologies considered. In case of delegation, however, the total execution time of the algorithm is driven by the second phase of the algorithm. Since this phase requires interaction between physically distant entities, its execution time may be significantly longer than the first phase (this involving only local actions). Several factors may affect the actual time requires for the second phase, such as the structure of the in-network overlay (INO), the number of neighbours in the INO, the physical distance between INO nodes, but also, the characteristics of the communication protocol to support the interactions. This section analyses how the two models proposed in Section (5) (i.e. the full-mesh model and the ring model) to organise the source nodes in the INO may affect the performance of DACoRM, both in terms

of convergence time and in terms of overhead associated with coordination among the nodes.

### 6.2.1 Experiments setup

The experiments performed rely on the computing laboratory facility of the Electronic and Electrical Engineering Department, where a set of physical computers connected to the Department's local network are available. The objective of the experiments is to analyse the evolution of the convergence time and the overhead incurred by the delegation process according to the number of nodes in the INO. In order to obtain results that are consistent with the characteristics of the adaptation process used in DACoRM, an specific experimental protocol have been designed.

Several set of experiments are performed by varying the number of nodes in the INO and the organisational model of the nodes, i.e. in full-mesh or in a ring. More precisely, for each experiment, one of the two models is chosen and a subset of the machines available in the laboratory is randomly selected. Each of the selected machine represents a single node and is identified by its IP address. The machines are then logically connected according to the model chosen through TCP socket connections.

The experimental protocol is then as follows. Initially one of the machines in the selected subset is randomly chosen to be the Deciding Entity node. This is responsible for initiating a communication with its neighbour according to the communication protocol designed to support the delegation process. According to the communication protocol, the Deciding Entity node sends a *COMPUTE_REQUEST* message to its neighbours. Upon receiving the request, each neighbour node needs to reply with a *COMPUTE_REPLY* message. In order to fill the different fields of the message, each node (or computer) is initially given a Sample File in which pre-recorded typical replies are provided. The file contains in particular the two possible status (fail or succeed) that can be reported by the neighbour nodes in practice and some examples of typical information to append. Each node is responsible for randomly selecting information to copy in the reply message from this file. It is worth noting that this action does not affect the actual outcome of the experiment, and is used only to enhance the scenario with more realistic messages. Before the reply is sent back to the Deciding Entity, a delay of 10 ms is artificially introduced, so that it represents the time normally required for a source node to execute the re-configuration algorithm locally and obtains a local solution. Upon receiving the different replies, the Deciding Entity then selects randomly one of the solution and sends the *APPLY_REQUEST* message to the relevant neighbour (which, in turn, acknowledge the request). The same scenario is then repeated iteratively 50 times.

The protocol designed to perform the experiments is thus consistent with adaptation process characteristics and the parameters used for these experiments are consistent with those considered to evaluate the performance of the adaptive scheme in the GEANT and the Abilene networks. More specifically, each experiment represent one cycle of the adaptation process where the number of iterations is equal to 50 and delegation is triggered at each iteration. It is important to note that although delegation may not be triggered at each iteration in a realistic scenario (see Section (6.1.7)), the evaluation considers the worst case scenario. In order to measure the relevant parameter of the communication protocol, all the events triggered during the experiment are monitored and copied in a log file, so that at the end of the experiment, the following information is analysed:

- the minimum time interval required between two successive iterations of the scenario ($T_{min}$). This interval is defined as the time between the selection of the

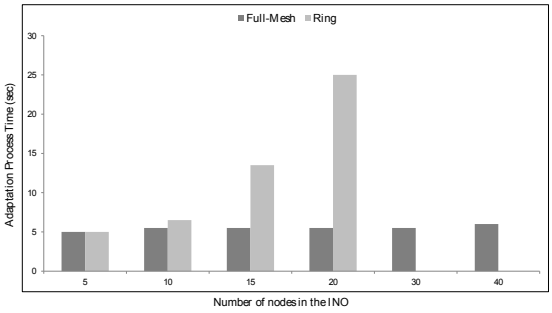


Figure 6.5: Evolution of the total execution time

Deciding Entity and the reception at the Deciding Entity of the acknowledgement from the node selected to enforce its solution.

- the total time required to complete the experiment ($T_{adaptation}$), i.e. to execute the scenario 50 times.
- the number of messages required during the all duration of the experiment.

Reported to the actual adaptation process, it is thus possible to evaluate the following:

- the minimum time interval between two successive iterations of the re-configuration algorithm during the adaptation process (through $T_{min}$).
- the total time required to complete a cycle of the adaptation process, i.e. to find and enforce new configurations (through $T_{adaptation}$).
- the volume of coordination messages required during the all duration of the experiment adaptation.

### 6.2.2 Execution time evolution

Figure (6.5) shows the evolution of $T_{adaptation}$ according to the number of nodes in the INO for the two models. It can be observed that the total time is not affected by the number of nodes in the full-mesh model, whereas this substantially grows as the number of nodes increases in the ring model. The results also show that the full-mesh model performs better than the ring model in terms of execution time. In fact, the ring model performs as well as the full-mesh for a small number of nodes (up to 10) but shows poor performance with a large number of nodes. Given the poor scalability performance achieved from only 20 nodes in this model, the experiments are not extended to a larger number of nodes.

To further analyse the influence of the number of nodes in the INO, the evolution of the minimum time interval between two successive iteration of the re-configuration algorithm during the adaptation process is determined. The evolution is shown in Figure (6.6).

The minimum interval is given by the minimum time required for completing all the steps of the delegation protocol, and especially for the nodes to receive all the expected messages. As it can be observed, the evolution of the minimum time interval is consistent with the evolution of the total execution time of the adaptation process. The time interval is not affected by the number of nodes in the full-mesh model, but it increases with the number of nodes in the ring model. Even if the actual time required for enabling communication between the different entities may be affected by

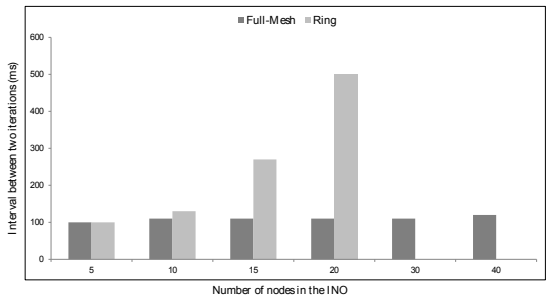


Figure 6.6: Evolution of the minimum interval between two consecutive iteration during the adaptation process

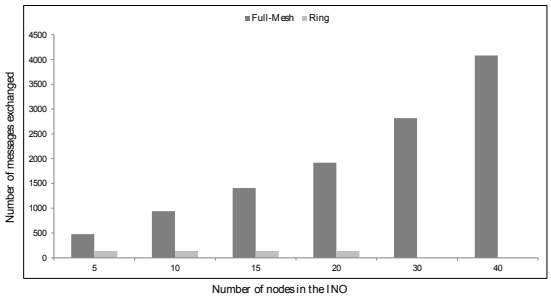


Figure 6.7: Evolution of the total number of coordination messages exchanged during the Adaptation Process

the physical distance between source nodes, as reported in [51][50], these results show that the total time required for the adaptation can be kept to an insignificant level (few seconds) compared to the frequency at which the adaptive resource management scheme is invoked, i.e. every 15 minutes.

### 6.2.3 Communication overhead evolution

The evolution of the total number of coordination messages exchanged during the adaptation process is presented in Figure (6.7) As explained previously, the worst case scenario is used for the experiment where delegation is triggered at each iteration of the adaptation process. It can be observed that the actual gap between the number of exchanged messages in the two models increases significantly as the number of nodes in the INO increases. These results show that the ring model scales better than the full-mesh model in terms of communication overhead. Given the poor scalability performance achieved in terms of delay, experiments are not performed with more than 20 nodes in the ring model.

It has been shown in Section (5.3.4) that the number of coordination messages required for each re-configuration interval in case of delegation is theoretically equal to $2N$ messages in the full-mesh approach and to 3 messages in the ring (where $N$ is the number of source nodes in the INO). Although the number of messages is independent of the number of INO nodes in the ring model, it linearly increases with the number of INO nodes in the mesh approach. To minimise the number of signalling messages exchanged, compute requests can be sent only to a limited number of neighbours, but this is at the risk of decreasing the probability of discovering a node that can perform a successful re-configuration. Given the small size of coordination messages (typically

less than 10 bytes), the overhead incurred by the delegation process is not significant given today's network capacities.

# Chapter 7

# Conclusion and Future works

## 7.1 Conclusion

Today's practices for managing network resources rely mainly on off-line traffic engineering approaches where the expected demand is calculated from previous usage and a specific routing configuration is produced, aiming to optimise resource usage for the next provisioning period. Given their static nature, these off-line approaches can be well sub-optimal in the face of changing or unpredicted traffic demand.

To overcome the limitations of current approaches, there have been some proposals for adaptive traffic engineering, whereby feedback information from the network is used to periodically adapt the routing configurations according to current network conditions. Despite these recent proposals, network resource management normally relies on centralised managers that periodically compute new configurations according to dynamic traffic behaviours. To meet the requirements of emerging services and applications, network resource management functionality that is decentralised, flexible, reactive and adaptive to traffic and network dynamics is necessary.

In this work, a new adaptive resource management scheme for intra-domain traffic engineering is proposed, so that the distribution of traffic in the network is controlled in an adaptive and decentralised manner according to network conditions. Unlike off-line traffic engineering schemes, which rely on static configurations, the proposed approach can efficiently deal with network and traffic dynamics by performing adaptations of routing configurations in short timescales.New configurations are not computed by a centralised management entity, but instead, are the results of re-configuration actions performed by the network source nodes, which coordinate among themselves through an in-network overlay to decide upon the most appropriate course of actions to follow in order to re-balance the traffic load.

The core chapters of this report focus on describing and analysing the main features and mechanisms of the proposed resource management approach. In order to demonstrate the performance of the proposed scheme, the different mechanisms used have been evaluated. The results of the evaluation indicate that a substantial gain in terms of resource utilisation can be achieved in the network with the proposed approach. In addition, it was shown that this near-optimal performance in terms of resource utilisation can be achieved in only few seconds, and this, without overloading the network with excessive coordination messages.

## 7.2 Future works

Several research topics will be investigated as future extension of the work presented in this report. The combination of the different parts of the current and future works will

build up the core chapters of the PhD thesis.

**Selection of the Deciding Entity** In order to limit the communication overhead incurred by the coordination between the source nodes (i.e. by the delegation process), the approach is designed in such a way that new configuration decisions are taken as much as possible locally by the selected Deciding Entity.

The rule used to select an unique Deciding Entity source node relies only on static and a priori computed information related to the virtual topologies and paths between each source-destination pair of nodes. As such, the selection of a Deciding Entity node does not take into account the effective capability of the node to compute new configuration.

The main advantages of this selection procedure is that each node can independently determine whether or not it can assume the role of the Deciding Entity for the current reconfiguration interval, without requiring any communication with the other nodes in the overlay. However, as shown and explained in Chapter (6), this selection procedure has some limitations highlighted by the analysis of the delegation frequency. In addition, whereas the Deciding Entity is responsible for selecting the b̈estsolution among the replies received from neighbour nodes in case of delegation, it does not consider other possible solutions in case a local solution can be determined. As such, this local solution may not be the optimal one.

In future work, a new selection procedure that relies on both static and dynamic information will be investigated. In a similar fashion to the current selection procedure, this needs to guarantee that only one source node at a time is selected to be the Deciding Entity, but this also needs to ensure that the selected node can effectively find a new configuration. In order to support this procedure, a scenario where each source node can determine the gain in the network that can be obtained with its local solution will be analysed. Based on the knowledge of the different gains, the node with the highest gain can be selected to be the Deciding Entity.

The main drawbacks of this approach compared to the previous one is that all source nodes need to communicate with each others in order to determine the highest gain, which may incur a significant communication overhead. To limit the number of coordination messages, other metrics will also be taken into account (such as static characteristics used in the previous case, or definition of some thresholds etc.), so that only a limited number of source nodes are expected to be involved in the selection process.

**In-network overlay structure** In order to effectively demonstrate how the delegation process can be supported, two simple organisational models for the in-network overlay are presented and analysed in this report.

Although initial experimental results are encouraging, it has been shown that these two models have some limitations as the number of nodes in the overlay increase (i.e. number of messages in the full-mesh model and total execution time in the ring model). In Chapter (5), an hybrid approach that can combine the advantages of the two previous models is discussed. In future extension of this work, the hybrid model will be further analysed and its performance will be evaluated over a large number of nodes in order to evaluate the scalability of the proposed approach for larger scale network topologies.

In addition, further analytical and experimental analysis of the ring model will be performed in order to determine in which configuration or case it may be used (for instance, maximum number of nodes, total delay to traverse the ring etc.).

Furthermore it would be interesting to enhance the hybrid model in such a way that any node in a local ring can directly communicate with the other rings in the net-

work, without transiting through an intermediate node. This may involve an additional management complexity that will be evaluated.

**Towards other in-network self-management applications** The third main extension of this work will focus on identifying generic patterns and infrastructure that can be used in different in-network adaptive resource management applications. In particular future work will investigate an approach to support in-network adaptive caching functionality, whereby nodes in the network equipped with caching capabilities coordinate among themselves to decide on the caching strategy to adopt in order to optimise content dissemination and resource usage efficiency. This part of the PhD work will also be interesting in analysing and modelling how the two applications can gracefully co-exist in the network, such as decisions related to each application do not lead to inconsistent network configurations. The interaction between the different control process will then be evaluated.

# Appendix A

# Evolution of the maximum utilisation in the network

The nature of the sequence of the maximum utilisation in the network at each iteration $(u_n^{max})_{n\in\aleph}$ is analysed. It is shown that the sequence $(u_n^{max})_{n\in\aleph}$ is monotonically decreasing, i.e.

$$u_{n+1}^{max} \leq u_n^{max} \tag{A.1}$$

**Proof** The notations used in the proof are similar with those used in Chapter (4). Without loss of generality suppose to be at iteration $n$ of the adaptation process. The utilisation of any link $l$ in the network is so that:

$$\forall l \in L u_n(l) < u_n^{max} \tag{A.2}$$

Let $F(S_i - D_j)$ be the traffic flow modified at the next iteration. Depending on its characteristics, a link $l$ in the network can be affected by the new configuration in six different ways as follows:

- **Case 1**: the link $l$ belongs to set $L \backslash S_{HU}$ but it is not used in any topologies to route the traffic flow that is modified. Its load is therefore not affected by the new configuration.
- **Case 2**: the link $l$ belongs to set $L \backslash S_{HU}$ and is used only in the topologies in the set $S_{l_{max}}^{S_i - D_j}$ to route the traffic flow that is modified. Since some traffic is moved away from these topologies, this results in a decrease in the load of link $l$.
- **Case 3**: the link $l$ belongs to set $L \backslash S_{HU}$ and is used only in the topologies in the set $\bar{S}_{l_{max}}^{S_i - D_j}$ to route the traffic flow that is modified. Since some traffic is moved towards these topologies, this results in an increase in the load of link $l$.
- **Case 4**: the link $l$ belongs to set $L \backslash S_{HU}$ and is used both in the topologies in the set $S_{l_{max}}^{S_i - D_j}$ and in the topologies in the set $\bar{S}_{l_{max}}^{S_i - D_j}$ to route the traffic flow that is modified. The variation of load of link $l$ depends on the difference between the volume of traffic diverted away and the volume of traffic received.
- **Case 5**: the link $l$ belongs to the set $S_{HU}$ According to the re-configuration algorithm, traffic cannot be diverted towards this link. However, it is used in the set $S_{l_{max}}^{S_i - D_j}$, so that some traffic is diverted away and the link load is decreased.
- **Case 6**: the link $l$ belongs to the set $S_{HU}$ and is not used in any of the topologies in the set $S_{l_{max}}^{S_i - D_j}$. Its load is therefore not affected by the new configuration.

The resulting link utilisation in the different cases is then as follows.

**Case 1** Since the link is not affected by the new configuration, $u_{n+1}(l) = u_n(l)$. In addition, by definition of the set $L \backslash S_{HU}$, $u_n(l) < u_n^{max}$. The utilisation of $l$ at iteration $n+1$ satisfies therefore $u_{n+1}(l) < u_n^{max}$.

**Case 2** The load of link $l$ is decreased by a factor $v \geq 0$, such as $\rho_{n+1}(l) = \rho_n(l) - v$. The utilisation of $l$ is therefore such as $u_{n+1}(l) = \frac{\rho_n(l)-v}{c(l)}$, i.e. $u_{n+1}(l) = u_{n+1}(l) - \frac{v}{c(l)}$. Since $v \geq 0$, the following is true: $u_{n+1}(l) < u_n(l)$. Since $l \in L \backslash S_{HU}$, the utilisation of $l$ at iteration $n+1$ satisfies $u_{n+1}(l) < u_n^{max}$.

**Case 3** The load of link $l$ is increased by a factor $v \geq 0$, such as $\rho_{n+1}(l) = \rho_n(l) + v$. The utilisation of $l$ is therefore such as $u_{n+1}(l) = \frac{\rho_n(l)+v}{c(l)}$, i.e. $u_{n+1}(l) = u_{n+1}(l) + \frac{v}{c(l)}$. Since $v$ satisfies the condition $v < V_{max}$, the utilisation of $l$ at iteration $n+1$ is therefore such as $u_{n+1}(l) < u_n^{max}$.

**Case 4** The load of link $l$ is increased by a factor $v \geq 0$ and decreased by a factor $v^{'} \geq 0$, such as $\rho_{n+1}(l) = \rho_n(l) + v - v^{'}$. The utilisation of $l$ is therefore such as $u_{n+1}(l) = \frac{\rho_n(l)+v-v^{'}}{c(l)}$, i.e. $u_{n+1}(l) = u_{n+1}(l) + \frac{v-v^{'}}{c(l)}$. Since $v$ satisfies the condition $v < V_{max}$ and $v^{'} \geq 0$, it can stated that $v - v^{'} < V_{max}$. And as such, the utilisation of $l$ at iteration $n+1$ satisfies $u_{n+1}(l) < u_n^{max}$.

**Case 5** This case is similar to case 2. Therefore $u_{n+1}(l) < u_n^{max}$.

**Case 6** This case is similar to case 1, except that the link belongs to the set $S_{HU}$. Therefore, $u_{n+1}(l) \leq u_n^{max}$.

Since any link $l$ in the network always falls within one of these cases, it can be stated that:

$$\forall l \in L \quad u_{n+1}(l) \leq u_n^{max} \tag{A.3}$$

It can be deduced from Equation (A.3), that $max_l(u_{n+1}(l)) \leq u_n^{max}$ and as a result, $u_{n+1}^{max} \leq u_n^{max}$, which shows that the sequence $(u_n^{max})_{n \in \aleph}$ is monotonically decreasing.

**End of proof**

# Bibliography


[1] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and Principles of Internet Traffic Engineering. RFC 3272 (Informational), May 2002. Updated by RFC 5462.

[2] A. Atlas and A. Zinin. Basic Specification for IP Fast Reroute: Loop-Free Alternates. RFC 5286 (Proposed Standard), September 2008.

[3] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer, IEEE*, 36:41–50, January 2003.

[4] An Architectural Blueprint for Autonomic Computing. June 2001.

[5] Jeffrey O. Kephart. Research challenges of autonomic computing. In *Proceedings of the 27th international conference on Software engineering*, ICSE '05, pages 15–22, New York, NY, USA, 2005. ACM.

[6] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle. A survey of autonomic network architectures and evaluation criteria. *Communications Surveys Tutorials, IEEE*, PP(99):1 –27, 2011.

[7] Steve R. White, James E. Hanson, Ian Whalley, David M. Chess, and Jeffrey O. Kephart. An architectural approach to autonomic computing. *Autonomic Computing, International Conference on*, 0:2–9, 2004.

[8] M. Muztaba Fuad and Michael J. Oudshoorn. System architecture of an autonomic element. In *Proceedings of the Fourth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems*, pages 89–93, Washington, DC, USA, 2007. IEEE Computer Society.

[9] Y. Maurel, Ph. Lalanda, and A. Diaconescu. Towards introspectable, adaptable and extensible autonomic managers. In *Proceedings of the 7th IEEE/IFIP International mini-Conference on Network and Service Management, Poster Session*, 2011.

[10] B. Jennings, S. van der Meer, S. Balasubramaniam, D. Botvich, M. O. Foghlu, W. Donnelly, and J. Strassner. Towards autonomic management of communications networks. *Communications Magazine, IEEE*, 45(10):112–121, October 2007.

[11] Simon Dobson, Spyros Denazis, Antonio Fernández, Dominique Gaïti, Erol Gelenbe, Fabio Massacci, Paddy Nixon, Fabrice Saffre, Nikita Schmidt, and Franco Zambonelli. A survey of autonomic communications. *ACM Trans. Auton. Adapt. Syst.*, 1:223–259, December 2006.

[12] R. Boutaba and J. Xiao. *Cognitive Networks: Towards Self-Aware Networks*, chapter Towards Self-Aware Networks, pages 77–95. Wiley, 2007.

[13] David D. Clark, Craig Partridge, J. Christopher Ramming, and John T. Wroclawski. A knowledge plane for the internet. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 3–10, New York, NY, USA, 2003. ACM.

[14] Hajer Derbel, Nazim Agoulmine, and Mikaël Salaün. Anema: Autonomic network management architecture to support self-configuration and self-optimization in ip networks. *Computer Networks*, 53:418–430, February 2009.

[15] A. Tizghadam and A. Leon-Garcia. Aorta: Autonomic network control and management system. In *INFOCOM Workshops 2008, IEEE*, pages 1 –4, april 2008.

[16] E. Lavinal, T. Desprats, and Y. Raynaud. A generic multi-agent conceptual framework towards self-management. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pages 394 –403, april 2006.

[17] S. Van der Meer, W. Donnelly, J. Strassner, B. Jennings, and M.O. FoghlÃº. Emerging principles of autonomic network management. In *Proceedings of the 1st IEEE International Workshop on Modelling Autonomic Communications Environments (MACE 2006)*, pages 29–48, 2011.

[18] N. Samaan and A. Karmouch. Towards autonomic network management: an analysis of current and future research directions. *Communications Surveys Tutorials, IEEE*, 11(3):22 –36, quarter 2009.

[19] O. Bonaventure, P. Trimintzios, G. Pavlou, B. Quoitin, A. Azcorra, M. Bagnulo, P. Flegkas, A. Garcia-Martinez, P. Georgatsos, L. Georgiadis, C. Jacquenet, L. Swinnen, and S. Tandel. *Quality of Future Internet Services, Cost263 final report*, chapter Internet Traffic Engineering, pages 118–179. Number 2856 in LNCS. Springer-Verlag, September 2003.

[20] Ning Wang, Kin Ho, G. Pavlou, and M. Howarth. An overview of routing optimization for internet traffic engineering. *Communications Surveys Tutorials, IEEE*, 10(1):36 –56, quarter 2008.

[21] D. Mitra and K.G. Ramakrishnan. A case study of multiservice, multipriority traffic engineering design for data networks. In *Global Telecommunications Conference, 1999. GLOBECOM '99*, volume 1B, pages 1077 –1083 vol. 1b, 1999.

[22] B. Fortz and M. Thorup. Internet traffic engineering by optimizing ospf weights. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 519 –528 vol.2, 2000.

[23] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional ip routing protocols. *Communications Magazine, IEEE*, 40(10):118 – 124, oct 2002.

[24] A. Sridharan, R. Guerin, and C. Diot. Achieving near-optimal traffic engineering solutions for current ospf/is-is networks. *Networking, IEEE/ACM Transactions on*, 13(2):234 – 247, april 2005.

[25] D. Xu, M. Chiang, and J. Rexford. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. *Networking, IEEE/ACM Transactions on*, PP(99):1, 2011.

[26] Simon Fischer, Nils Kammenhuber, and Anja Feldmann. Replex: dynamic traffic engineering based on wardrop routing policies. In *Proceedings of the 2006 ACM CoNEXT conference*, CoNEXT '06, pages 1:1–1:12, New York, NY, USA, 2006. ACM.

[27] A. Elwalid, C. Jin, S. Low, and I. Widjaja. Mate: Mpls adaptive traffic engineering. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1300 –1309 vol.3, 2001.

[28] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. Walking the tightrope: responsive yet stable traffic engineering. *SIGCOMM Comput. Commun. Rev.*, 35:253–264, August 2005.

[29] Ivan Gojmerac, Peter Reichl, and Lasse Jansen. Towards low-complexity internet traffic engineering: The adaptive multi-path algorithm. *Comput. Netw.*, 52:2894–2907, October 2008.

[30] Ning Wang, Kin-Hon Ho, and George Pavlou. Adaptive multi-topology igp based traffic engineering with near-optimal network performance. 4982:654–666, 2008. 10.1007/978-3-540-79549-0:57.

[31] C. Villamazir. Ospf optimized multipath (ospf-omp), 1999.

[32] J. Moy. OSPF Version 2. RFC 2328 (Standard), April 1998. Updated by RFC 5709.

[33] A. Kvalbein, C. Dovrolis, and C. Muthu. Multipath load-adaptive routing: putting the emphasis on robustness and simplicity. In *Network Protocols, 2009. ICNP 2009. 17th IEEE International Conference on*, pages 203 –212, oct. 2009.

[34] P. Casas, F. Larroca, J.-L. Rougier, and S. Vaton. Robust routing vs dynamic load-balancing a comprehensive study and new directions. In *Design of Reliable Communication Networks, 2009. DRCN 2009. 7th International Workshop on*, pages 123 –130, oct. 2009.

[35] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault. Multi-Topology (MT) Routing in OSPF. RFC 4915 (Proposed Standard), June 2007.

[36] T. Przygienda, N. Shen, and N. Sheth. M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs). RFC 5120 (Proposed Standard), February 2008.

[37] S. Gjessing. Implementation of two resilience mechanisms using multi topology routing and stub routers. In *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, page 29, feb. 2006.

[38] M. Menth and R. Martin. Network resilience through multi-topology routing. In *Design of Reliable Communication Networks, 2005. (DRCN 2005). Proceedings.5th International Workshop on*, pages 271 – 277, oct. 2005.

[39] A. Kvalbein, A.F. Hansen, T. Cicic, S. Gjessing, and O. Lysne. Multiple routing configurations for fast ip network recovery. *Networking, IEEE/ACM Transactions on*, 17(2):473 –486, april 2009.

[40] Jun Wang, Yaling Yang, Li Xiao, and Klara Nahrstedt. Edge-based traffic engineering for ospf networks. *Computer Networks*, 48:605–625, July 2005.

[41] Amund Kvalbein and Olav Lysne. How can multi-topology routing be used for intradomain traffic engineering? In *Proceedings of the 2007 SIGCOMM workshop on Internet network management*, INM '07, pages 280–284, New York, NY, USA, 2007. ACM.

[42] Srikanth Sundaresan, Cristian Lumezanu, Nick Feamster, and Pierre Francois. Autonomous traffic engineering with self-configuring topologies. In *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM, Short Paper*, SIGCOMM '10, pages 417–418, New York, NY, USA, 2010. ACM.

[43] Matthew Caesar, Martin Casado, Teemu Koponen, Jennifer Rexford, and Scott Shenker. Dynamic route recomputation considered harmful. *SIGCOMM Comput. Commun. Rev.*, 40:66–71, April 2010.

[44] M. Laor and L. Gendel. The effect of packet reordering in a backbone link on application throughput. *Network, IEEE*, 16(5):28 – 36, sep/oct 2002.

[45] Steve Uhlig, Bruno Quoitin, Jean Lepropre, and Simon Balon. Providing public intradomain traffic matrices to the research community. *SIGCOMM Comput. Commun. Rev.*, 36:83–86, January 2006.

[46] John Risson and Tim Moors. Survey of research towards robust peer-to-peer networks: search methods. *Computer Networks*, 50:3485–3521, December 2006.

[47] Eng Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys Tutorials, IEEE*, 7(2):72 – 93, quarter 2005.

[48] D. Katz, K. Kompella, and D. Yeung. Traffic Engineering (TE) Extensions to OSPF Version 2. RFC 3630 (Proposed Standard), September 2003. Updated by RFCs 4203, 5786.

[49] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFCs 1122, 3168, 6093.

[50] Han Zheng, Eng Keong Lua, Marcelo Pias, and Timothy G. Griffin. Internet routing policies and round-trip-times. In *PAM*, pages 236–250, 2005.

[51] Yong Zhu, C. Dovrolis, and M. Ammar. Combining multihoming with overlay routing (or, how to be a better isp without owning a network). In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 839 –847, may 2007.

[52] T. Klingberg and R. Manfredi. Gnutella Protocol Development, Gnutella v0.6, September 2002.

[53] S. Balon, J. Lepropre, O. Delcourt, F. Skivee, and G. Leduc. Traffic engineering an operational network with the totem toolbox. *Network and Service Management, IEEE Transactions on*, 4(1):51 –61, june 2007.

[54] The abilene topology and traffic matrices dataset, 2004. http://www.cs.utexas.edu/ yzhang/research/AbileneTM/.

[55] The geant topology, 2004. http://www.dante.net/server/show/nav.007009007.

[56] Totem project: Toolbox for traffic engineering methods. http://totem.run.montefiore.ulg.ac.be/.