

Project Number : IST-1999-11253-TEQUILA

Project Title : Traffic Engineering for Quality of Service in the Internet, at Large Scale



D3.4: Final System Evaluation

Part C: Scalability and Stability Analysis

CEC Deliverable Nr: 304/Algonet/b1

Deliverable Type: Report

Deliverable Nature: Public

Contractual date: October 30, 2002

Actual date: October 30, 2002

Editor : Takis Damilatis

Contributors :

Alcatel: Danny Goderis, Geoffry Crystallo

Algosystems: P. Georgatsos, T. Damilatis, M.Megalooikonomou

FT-R&D: C. Jacquenet, M.Boucadair, C. Rischette, C. Duret, J. Lattmann, V. Laspreses

IMEC: S. Van Den Berghe, Pim Van Heuven

NTUA: Eleni Mykoniati, D. Giannakopoulos, M. Maurogiorgis

Global Crossing (Thales): H. Asgari, M. Irons, R. Egan

UCL: D. Griffin

UniS: P. Trimintzios, P.Flegkas, G.Pavlou

Workpackage(s) : WP3

Abstract : This Deliverable includes the results and conclusions obtained by the integration, component performance, and system performance tests carried out in testbeds and simulators.

Keyword List : DiffServ, Traffic Engineering, Monitoring, Measurement, Router, Service Level Specification, Policy Management, Test Suite, Test Purpose, Experimentation.

Project Number : IST-1999-11253-TEQUILA

Project Title : Traffic Engineering for Quality of Service in the Internet, at Large Scale



D3.4: Final System Evaluation

Part C: Scalability and Stability Analysis

Editor: Takis Damilatis

Contributors:

Alcatel: Danny Goderis, Geoffry Crystallo

Algosystems: P. Georgatsos, T. Damilatis, M.Megalooikonomou

FT-R&D: C. Jacquenet, M.Boucadair, C. Rischette, C. Duret, J. Lattmann, V. Laspreses

IMEC: S. Van Den Berghe, Pim Van Heuven

NTUA: Eleni Mykoniati, D. Giannakopoulos

Global Crossing (Thales): H. Asgari, M. Irons, R. Egan

UCL: D. Griffin

UniS: P. Trimintzios, P.Flegkas, G.Pavlou

Version: Final

Date: October 30, 2002

Distribution: WP3

© Copyright by the TEQUILA Consortium

The TEQUILA Consortium consists of:

Alcatel	Coordinator	Belgium
Algonet S.A.	Principal Contractor	Greece
FT-CNET	Principal Contractor	France
IMEC	Principal Contractor	Belgium
NTUA	Principal Contractor	Greece
Global Crossing (Thales Research)	Principal Contractor	United Kingdom
UCL	Principal Contractor	United Kingdom
TERENA	Assistant Contractor	The Netherlands
UniS	Principal Contractor	United Kingdom

Executive Summary

This document is the Part C Deliverable D3.4 and provides an analysis of the TEQUILA system with respect to scalability and stability. The behaviour of the TEQUILA system is exercised at a theoretical level and potential bottlenecks and sources of instabilities (control loops) are identified.

Scalability and stability have been of primary concern to the TEQUILA system, right from its very design. Appropriate design principles and mechanisms have been employed to cope with scalability and stability. These are also discussed.

By its very nature, the problem of QoS provisioning bears scalability and stability issues. A valid QoS provisioning solution should scale with the entities of the environment (customers/users, network) and should operate safely, without oscillations and instabilities, converging to acceptable levels of network operation. As such, QoS solutions based on per flow signalling/reservations in the network (e.g. IntServ-based) may suffer from scalability and QoS solutions relying on continuous optimising (fine-grained dynamic) traffic engineering (TE) schemes may experience instabilities (let alone the incurred overhead). On the other hand, QoS-solutions building on aggregate flows, rather than individual or streams of flows, and relying on coarse TE schemes, operating at medium to longer time dynamics, seem viable. However, flow aggregation and coarseness comes at the expense of accuracy and per flow guarantees. Therefore, there is a clear trade-off between the performance and the efficiency of QoS provisioning solutions.

The problem of scalability and stability of QoS provisioning solutions is even more complex considering the very nature of the telecommunications environment. The unpredictable nature of user traffic and the nature of the IP traffic itself (e.g. elastic TCP/IP-based traffic) constitute a major source of potential instability, increasing also the complexity and overhead of QoS solutions. Furthermore, the multi-service and the SLA-based nature of QoS offerings increase the number and volume of the entities to be handled, therefore influencing the scalability of QoS solutions.

The document is organised as follows. Section 1 offers an overview of the TEQUILA system, emphasising on aspects relevant to scalability and stability. Section 2 discusses overall system considerations regarding scalability and stability and introduces the terminology of the entities influencing system behaviour. Sections 3 and 4 present the scalability and stability analysis, respectively. Section 5 presents the main results of the analysis.

Table of Contents

1	OVERVIEW OF THE TEQUILA SYSTEM FROM SCALABILITY AND STABILTY POINT OF VIEW	6
1.1	RATIONAL, PRINCIPLES AND NOTIONS.....	6
1.1.1	<i>A Hierarchical Model for QoS IP Services</i>	6
1.1.2	<i>Service-driven TE</i>	6
1.1.3	<i>Two-level Service Admission Control</i>	8
1.2	THE TEQUILA FUNCTIONAL MODEL	9
2	SCALABILITY AND STABILITY CONSIDERATIONS AND TERMINOLOGY	12
2.1	OVERALL SYSTEM CONSIDERATIONS	12
2.2	TERMINOLOGY	13
3	SCALABILITY ANALYSIS	14
3.1	INTRODUCTION	14
3.2	RPC EPOCH.....	14
3.2.1	<i>Traffic Forecasting</i>	14
3.2.2	<i>Network Dimensioning</i>	16
3.3	SUBSCRIPTION EPOCHS	19
3.4	INVOCATION EPOCHS.....	21
3.5	DYNAMIC EPOCHS.....	23
3.5.1	<i>RSIM-Dynamic Admission Management</i>	23
3.5.2	<i>Dynamic Route Management</i>	24
3.5.3	<i>Dynamic Resource Management</i>	26
3.6	MONITORING SYSTEM	27
4	STABILITY ANALYSIS	30
4.1	INTRODUCTION	30
4.2	CONVERGENCE OF ALGORITHMS/PROTOCOLS	30
4.2.1	<i>Network Dimensioning Algorithms/Protocols</i>	30
4.2.2	<i>Service Admission Algorithms/Protocols</i>	30
4.2.3	<i>Dynamic TE Algorithms/Protocols</i>	31
4.3	IDENTIFICATION OF CONTROL LOOPS.....	31
4.4	STABILITY WRT. CONVERGENCE OF OPERATIONS.....	31
4.4.1	<i>Issues and Requirements</i>	31
4.4.2	<i>Considerations</i>	32
4.4.3	<i>Dynamic Admission Management Control-Loop</i>	33
4.5	STABILITY WRT. OSCILLATIONS	34
5	CONCLUSIONS.....	35
6	APPENDIX A – ANALYSIS OF CONSTRAINED STEINER TREE ALGORITHMS	38
7	REFERENCES.....	39

List of Figures

Figure 1: The TEQUILA Model for Internet QoS Services.....	6
Figure 2: TEQUILA Functional Model.	10

1 OVERVIEW OF THE TEQUILA SYSTEM FROM SCALABILITY AND STABILITY POINT OF VIEW

1.1 Rational, Principles and Notions

1.1.1 A Hierarchical Model for QoS IP Services

The essence of the TEQUILA system design is two-fold:

- Specify what are QoS-based IP connectivity services.
- Adopt a holistic view for providing QoS-based services; vertically, from QoS services to PHBs, and horizontally, from service request handling to service fulfilment and service assurance.

TEQUILA has proposed [D1.1], a well-defined template for describing the technical characteristics of QoS-based IP connectivity services, generally referred to as Service Level Specifications (SLSs). Considering that connectivity services may comprise several ‘connectivity legs’, the *TEQUILA SLS template* describes the technical characteristics (topology, IP flows, transfer quality characteristics, compliance criteria) of a single, unidirectional ‘connectivity leg’. A connectivity service then, is a collection of such SLS templates, bound to the same customer and the same access, usage means and characteristics. The benefits of a standard, in the sense of widely approved, service description template, as the one proposed by TEQUILA, include: automation of service provisioning cycle and facilitation of service provisioning in the Internet.

In establishing a holistic view for QoS service provisioning, there is a missing link between the concept of PHB, the basic QoS building block in IP DiffServ, and QoS-based services. TEQUILA has filled this gap by introducing the notion of *QoS-class*. A QoS-class consists of an Ordered Aggregate (or PHB Scheduling Class, e.g. AF1) and associated QoS parameter(s) such as one-way transit delay, inter-packet delay variation or packet loss.

QoS-classes depict the elementary QoS transfer capabilities of an SP domain, between SP edges. They are not services per se; instead services are built on them. The concept of QoS-class can be compared to the Per Domain Behaviours (PDBs), some of which are currently being specified by the DiffServ WG of the IETF. Figure 1 summarises the above, presenting the TEQUILA layered model for IP QoS services.

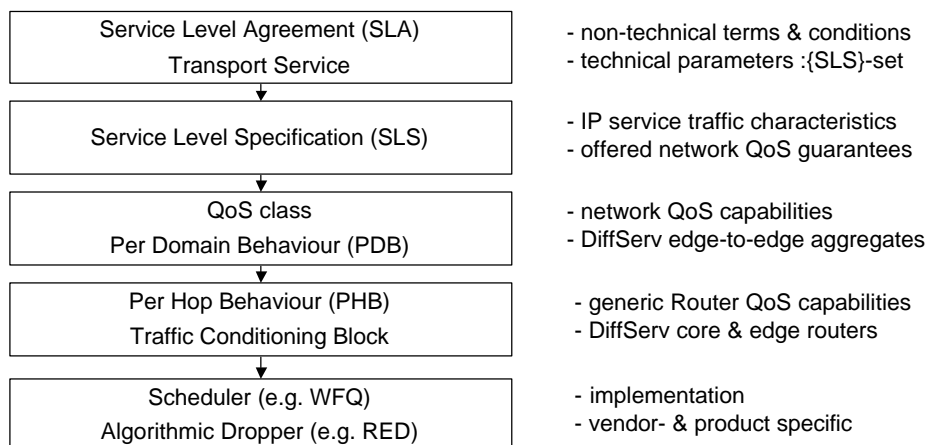


Figure 1: The TEQUILA Model for Internet QoS Services.

1.1.2 Service-driven TE

TEQUILA clearly distinguishes between *service and resource layers*. The resource layer employs network management and traffic engineering (TE) functions (the latter being of concern to TEQUILA), while the service layer includes the necessary functions for offering/establishing service agreements (based on the TEQUILA SLS template) and subsequently handling the admission of service requests.

Although distinguished, the service layer and TE functions in the TEQUILA system *do not act in isolation*. TE functions provide the grounds on which the service layer functions operate and, conversely, the service layer sets the traffic-related objectives [TE-FRAM] of the TE functions to achieve. More specifically, in the TEQUILA system, the service layer provides TE functions with the *Traffic Matrix (TM)*, which specifies anticipated QoS traffic demand between network edges. Traffic demand is forecasted from historical data and/or SP expectations (e.g. sales targets). Based on these traffic forecasts, the network is appropriately *dimensioned (off-line engineered)* by the TE functions, in terms of PHBs and their configuration parameters and QoS route constraints. In turn, the TE functions provide the service layer functions with the *Resource Availability Matrix (RAM)*, which specifies estimates of the availability of the engineered network to accommodate QoS traffic between network edges. Based on these availability estimates, the service layer functions, also utilising network-state information (see below), handle the admission of QoS-based service requests so that not to overwhelm the network.

As scalability and convergence are of primary concern, interactions and information exchanged between service layer and TE functions should not be (too) fine-grained and should rely on aggregated information in order to avoid oscillations and high overhead. As such, the service layer and the TE functions in the TEQUILA system exchange information (TM, RAM), which refers to *Traffic Trunks (TTs)*, not individual customer contracts or flows, at the granularity of *Resource Provisioning Cycles (RPCs)*.

A Traffic Trunk is a QoS-class (cf. previous section) in a certain topological scope with respect to a particular domain. That is, TTs are aggregates of QoS traffic having the transfer characteristics of the associated QoS-class over customers' contracts between network edges. It should be noted that TTs span between SP edges; not between distinct destinations in the Internet, nor between distinct customer sites. For example, for a given SP domain, AF1-like quality Internet traffic corresponds to as many TTs as the combination of the edges of the SP and the available Internet gateways in the domain. In TEQUILA, TTs may be of either pipe (point-to-point) or hose (point-to-multi-point) topological scope. The latter are useful for covering the needs of services like multicast and VPNs.

A Resource Provisioning Cycle is a period of time during which anticipated QoS traffic demand, for the supported TTs, is believed to be valid. Should anticipated QoS traffic demand prove no longer valid, a new RPC is initiated. New RPCs may also be initiated when the service layer or the TE functions realise, each from its own perspective, that they can no longer gracefully provide the QoS actually being requested. TE functions realise that, when the network cannot longer *deliver the QoS targets of the supported QoS-classes*, and service layer functions when they cannot longer *satisfactorily sustain the actual offered load at the contracted QoS levels*. Note that RPCs may also be initiated by network administration (e.g. when new points of presence or resources are installed).

The interactions between the service layer and the TE functions as described previously (traffic, availability matrices) occur *only at RPC epochs*, not at the granularity of every single or a few service requests. That is, only when a new RPC commences, a traffic matrix is produced, the network is appropriately re-engineered and the resource availability matrix is in turn produced.

By their very definition, RPCs are mostly long time periods (days to months). RPCs are bound to traffic demand forecasts, which are usually drawn with long term perspectives. Moreover, as traffic demand forecasts refer to TTs, aggregating QoS traffic over customers' contracts between network edges, the likelihood of the demand forecasts being valid in a longer time period is increased (law of large numbers).

As anticipated QoS traffic demand and network availability estimates are forecasts, they should be treated by the TE and service layer functions as such. Actual offered traffic is to fluctuate around the forecasted values, some times even beyond acceptable statistical errors. TEQUILA interprets QoS traffic demand forecasts as nominal rather than actual values and network availability estimates as guidelines, rather than stringent directives.

On the above grounds, the TEQUILA system incorporates *dynamic service layer and TE functions*, which, by utilising actual network state and load information, see into optimising network performance in terms of service admissions and resource utilisation, while meeting traffic QoS constraints. Specifically, TEQUILA opts for dynamic functions for handling service admissions, managing network resources (PHBs) and routing. These dynamic functions operate during an RPC utilising the results produced by the TE functions - network dimensioning and availability results- at the beginning of the RPC as guidelines, mainly *for deriving appropriate QoS routes and for resolving congestion*.

As scalability and stability are of concern, the feedback control-loops of the dynamic functions are based on threshold crossing events based on moving averages rather than on polling instantaneous values. Furthermore, the monitored entities are of coarse nature (e.g. PHBs, LSPs, TTs).

The above approach for TE can be also referred to as “*first plan, then take care*”, whereby decisions, taken with long-term perspectives (beginning of RPCs), are refined (by dynamic functions) according to actual developments in shorter time periods. The initial decisions and their refinements are driven by clearly defined objectives; meet QoS requirements and optimise network performance. This approach combines the merits of centralised and distributed decision making schemes. It is not monolithic, as pure centralised schemes, while at the same time it harnesses the dynamics of state/load-dependent schemes on the basis of guidelines produced with longer-term vision. It is believed that the approach yields increased levels of control to network performance and also facilitates the enforcement of related policies; features, which are highly appreciated by SPs.

1.1.3 Two-level Service Admission Control

TEQUILA distinguishes two epochs for service requests: *subscription* and *invocation*. At subscription epochs, customers subscribe to services in order to gain the right to use these services later, as they require. At invocation epochs, the users of the customers invoke services to which they have subscribed. Services can be invoked either *explicitly*, using particular signalling means/protocols (e.g. RSVP) or *implicitly*, being active throughout the subscribed service schedule (e.g. leased-line VPN services). On successful service invocation, the corresponding traffic flows are generated in the users’ hosts and injected to the network. Based on the specified SLS template, TEQUILA has also specified templates for service subscriptions and invocations, called *SSS (Service Subscription Structure)* and *SIS (Service Invocation Structure)* respectively; these templates include multiple SLSs, as an SLS depicts one of the legs of a connectivity service.

The distinction between service subscription and invocation is backed up by a number of good reasons. It directly corresponds to current business models and practices, it is required for AAA (Authentication, Authorisation and Accounting), it enables business-driven service provisioning, facilitates traffic demand derivation and prompts for higher aggregation levels –at customer and/or customer-type levels, therefore increasing scalability.

The service layer in the TEQUILA system incorporates *admission logic at both subscription and invocation epochs*, with the purpose not to overwhelm the network with traffic beyond its current capacity to provide QoS. *Subscription logic* aims at ensuring that the network can indeed operate within acceptable levels with respect to the trade-off between number of subscriptions and confidence for delivering contracted QoS (as in SSS/SLSS). *Invocation logic* aims at ensuring that the network operates within acceptable levels with respect to the trade-off between volume of traffic and likelihood of network performance deterioration, that is to ensure that the QoS traffic to actually enter the network can be gracefully sustained, given network current status. In addition, to ensure that incoming traffic conforms to contractual requirements (as in SSS/SLSS).

Subscription logic operates during an RPC in a centralised fashion and utilises information regarding the availability of the engineered network to accommodate QoS services (cf. RAM), produced by the TE functions at the beginning of the RPC. Based on this information, and related policy parameters specified by TEQUILA [D1.3], for a requested subscription, the subscription logic determines whether to accept it as is, or to propose alternatives or to reject it. To enable automation of the subscription process, TEQUILA has specified a suitable negotiation protocol, the *Service Negotiation Protocol, (SrNP)* [D1.2].

Invocation logic operates during an RPC and is distributed at network edges. To increase its stability and minimise overhead, invocation logic relies on *network state on-off indicators*, rather than on network load measurements. Two on-off indicators are considered: *network green and network red*, indicating whether or not the network can sustain the performance of the supported QoS-classes, given current load. The indicators are 'lit' by the network monitoring and/or SLS assurance services. In addition, invocation logic utilises locally available information on QoS traffic currently being admitted; also, it uses the network availability estimates (RAM) produced at the beginning of the RPC as guidelines. Based on this information and related policy parameters specified by TEQUILA [D1.3], invocation logic determines the parameters of suitable *traffic actions* for handling *future and existing admissions* so that not to overwhelm the network. The traffic actions are: probability-based admission control schemes, which apply to newly requested invocations (in addition to the standard authentication and authorisation checks for ensuring compliance with subscription profiles), and the standardised Diffserv traffic conditioners -classifiers, markers and policers- which apply to the currently admitted flows. All these actions apply at the granularity of groups of SSS/SLSs (e.g. per customer type) and they are arranged in various severity levels, depending on the network-state indicators and the *pro-activeness level* with which invocation logic has been configured to operate (set by policies).

1.2 The TEQUILA Functional Model

Summarising the analysis of the TEQUILA QoS solution presented in the previous section, the TEQUILA system, at a high level, is decomposed into the following major sub-systems (Figure 2):

- *SLS Management system (SLS-Mgt)*, encompassing the service layer functions for SLS-based QoS service provisioning, being responsible for offering/agreeing with customers QoS services and handling respective requests.
- *Traffic Engineering (TE) system*, encompassing the TE functions of the resource layer, being responsible for fulfilling the contracted services, by appropriately engineering the network.
- *Monitoring system*, for providing the above systems with the appropriate network measurements and for assuring that the contracted services are indeed delivered at their QoS.

Recognising that service provisioning largely depends on business policies, the *Policy Management* system is introduced to compile business rules and practices to parameters understood by the previously introduced systems. This way, it would be possible to tailor the operation of the system to the specific business policies of the particular SP, increasing therefore system usability.

With reference to the functionality outlined and the terminology introduced in the previous section, the TEQUILA system can be further decomposed into the following functional components per sub-system, arranged per epochs of system operation.

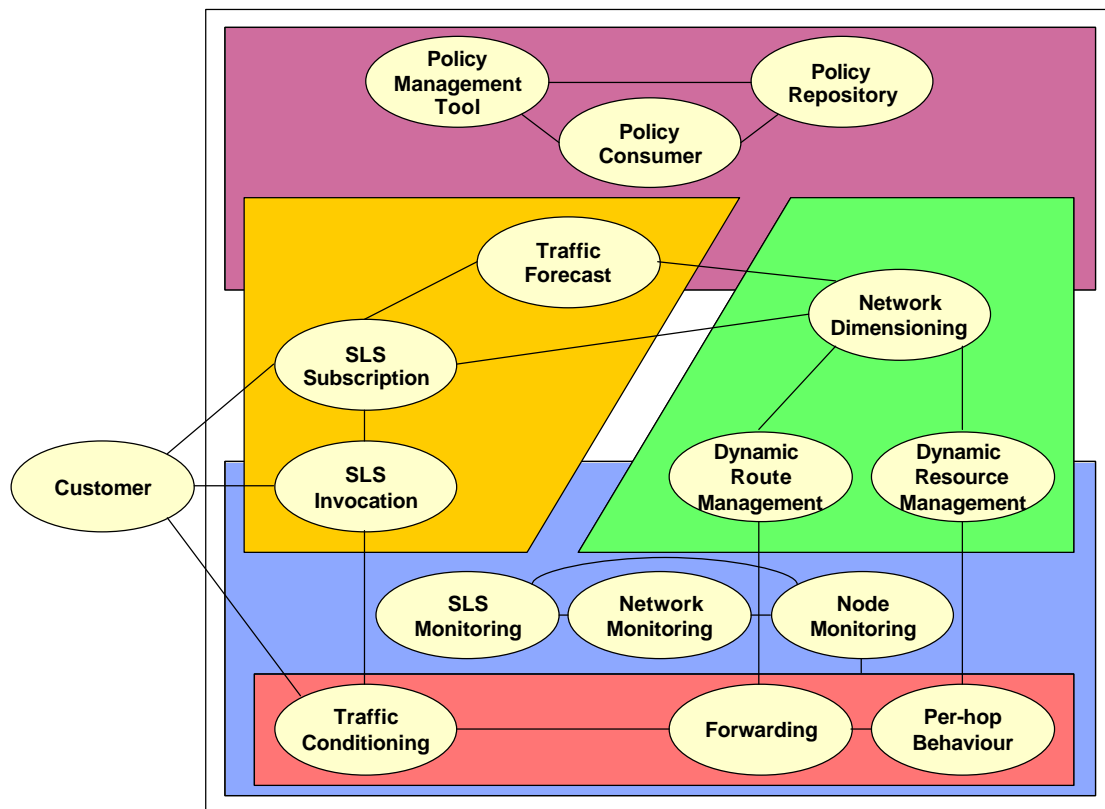


Figure 2: TEQUILA Functional Model.

RPC epoch components

Traffic Forecast Management (TFM), deriving the anticipated QoS traffic demand per TT based on historical data and business policies, and verifying its validity against actually offered traffic.

Network Dimensioning (ND), implementing the off-line TE functions for dimensioning the network based on the derived QoS traffic demand forecasts, also deriving the estimates of the availability of the engineered network for QoS traffic per TT.

Depending on the routing scheme employed in the network, TEQUILA has seen into an MPLS-based and a pure IP-based TE approach. The dimensioning process of ND results in the definition of PHB configuration parameters and QoS routing constraints. The latter, in the MPLS TE case are translated into LSPs and in the IP TE case into link weights per QoS-class. The dimensioning results are downloaded to the dynamic TE components (see below), which treat them as guidelines for routing and congestion resolution, and configure appropriately the routers according to network state and actual load conditions.

Subscription epoch components

Service Subscription Management (SSM), implementing the logic and the means for negotiating service subscriptions with the customers.

Invocation epoch components

Router Service Invocation Management (RSIM), implementing the logic and the means for handling service admissions. Note that this component has also state-dependent parts.

Dynamic components

Node, Network Monitoring (NodeMon, NetMon), realising the passive and active monitoring jobs, requested by other system components and providing the required measurements and threshold crossing events.

Dynamic Resource Management (DRsM), managing the scheduling parameters of the installed PHBs according to actual load conditions, taking into account the dimensioning guidelines produced by ND at the beginning of the RPC.

Dynamic Route Management (DRtM) -only in the MPLS TE approach-, managing the allocation of subscribed address groups to the defined LSPs according to actual load conditions, taking into account the dimensioning guidelines produced at the beginning of the cycle.

Enhanced OSPF Routing -only in the IP TE approach-, implementing an enhanced OSPF-based routing algorithm (DSCP-aware), according to the dimensioning results produced by ND at the beginning of the RPC.

2 SCALABILITY AND STABILITY CONSIDERATIONS AND TERMINOLOGY

2.1 Overall System Considerations

As already became evident from the previous chapter, the TEQUILA system has been built around design principles with inherent scalability and stability merits, summarized in the following:

- *Multiple levels of aggregation.* From individual customer flows to SLSs (per customer aggregation), to subscriptions (per customer/service-type aggregation) and to TTs (topological and QoS aggregation).
- *Timely and clear separation of responsibilities.* The TEQUILA system operates in distinct epochs in different time scales, executing functions of increasing complexity and finer abstractions as moving to higher time scales. From light-weight dynamic functions applying on aggregates, TTs, PHBs but requiring network load feedback, to service admission functions applying on per SLS and subscription aggregates and mainly utilizing local information, and finally to heavy processes, requiring no feedback from the network, running off-line at the beginning of RPCs.
- *No network load-dependent service admission control schemes.* A liberal two-level service admission control scheme is adopted. Subscription negotiation operating off-line at epochs of new subscriptions, utilizing readily available information produced off-line at the beginning of the cycle; combined with light-weight admission control operating at service invocation instances, mainly relying on local information and coarse network state indications.
- *No explicit resource reservations, no signalling in the network.* Bandwidth is allocated only to PHBs, by setting their associated link scheduling parameters. Bandwidth is not allocated to any other network resources (e.g. LSPs are used as plain route-maps), nor is reserved across the network, through appropriate signalling, for accommodating the QoS requirements of flows or their aggregates.
- *Distributed architecture.* The major part of the system is distributed, residing mainly at the edges (as the Diffserv architecture itself advocates) and at the core. No protocols for coordinating the operation and actions of the distributed components. Rather, they interact independently, relying on network state measurements.
- Feedback control loops are based on *smoothed (moving averages), not instantaneous measurements*, relying on *threshold crossing events, not on continuous polling*.
- *Hierarchical monitoring architecture.* Monitoring is done in a hierarchical (from node monitoring and associated agents, to network and SLS monitoring), distributed architecture, pushing the required polling and active probes (injected test traffic) close to the source of the data (network routers). Alternatively to active monitoring, edge-to-edge resource statistics (for LSP, SLSs) could be measured by aggregating hop-by-hop statistics.

The above features clearly contribute to system scalability and stability; however, at the expense of accuracy, granularity and abstraction.

2.2 Terminology

Let:

- $E, |E|$ be the set and the number of the network links, respectively; links are meant unidirectional
- $V, |V|$ be the set and the number of the network nodes, respectively
- $V_E, |V_E|$ be the set and the number of the network edges, respectively.
- $H, |H|$ be the set and the number of the QoS-classes (or equivalently Ordered Aggregates, cf. section 1.1.1), supported by the network, respectively.
- $SSS, |SSS|$ be the set and the number of the subscriptions established by the network, respectively.
- $SLS, |SLS|$ be the set and the number of the subscribed SLSs, respectively. Note that the number of SLSs is, in general, larger than the number of subscriptions, as a subscription may include a number of SLSs. It should be stressed that, as will become apparent in the analysis to follow, the number of SLSs and therefore the number of subscriptions is bound by the vendor-specific capabilities of the edge routers.
- $T, |T|$ be the set and the number of the TTs (cf. section 1.1.2) supported by the network, respectively. They present the basic commodities to be served by the network and therefore, the network needs to be appropriately traffic-engineered so that to efficiently accommodate their offered load.
- $T_P, |T_P|$ be the set and the number of the TTs of pipe (point-to-point) topological scope supported by the network, respectively. The order of magnitude of these TTs is $O(|V_E|^2 * |H|)$.
- $T_H, |T_H|$ be the set and the number of the TTs of hose (point-to-multi-point) topological scope supported by the network, respectively. The order of magnitude of these TTs is $O(2^{|V_E|} * |H|)$. As this number grows prohibitively with the number of edges, TEQUILA *poses limits on the number of hoses per edge* against which the network will be dimensioned. The limited number of hoses is created by topologically aggregating the different hose TTs requested in the SSS/SLSs.
- $S_T, |S_T|$ be the set and the number of trees found per TT, by the network dimensioning algorithm, when the forecasted QoS traffic demand is mapped into the network resources. The term ‘tree’ is used instead of the term ‘path’, to indicate that the TTs might be of hose topology, that is point-to-multi-point. These trees materialise the QoS routing constraints, in the sense that TT traffic can be accommodated only through these trees. As it will be seen later, the maximum worst case number of trees per TT is \bar{C} , the maximum number of iterations of the network optimisation algorithm, which is set by policies.
- $C_T, |C_T|$ be the set and the number of the address groups as specified in the SLSs, for all SLSs contributing to a TT.
- $Cust, |Cust|$ be the set and the number of customers having subscriptions, respectively.

3 SCALABILITY ANALYSIS

3.1 Introduction

For the purpose of scalability analysis, the system is viewed as a collection of interacting decision-making components. For each such component, we analyse

- the scaling and complexity of the decision making process itself -*decision making*, and
- the scaling and complexity of the produced output parameters -*decision implementation*

The decision making process is decomposed into operations (e.g. algorithm) and for each operation complexity analysis is provided as a function of external or internal system parameters, influencing the behaviour of the component under study. This way, the implied bottlenecks are identified and analysed. The convergence of the operations/algorithms is proved where applicable.

The analysis concentrates on a theoretical and high-level description of the expected *worst-case* behaviour. As is usually the case, the performance of the system in practice can be better than the worst-case behaviour. Hence the discussion in this section should be complemented by experimental results in order to get a more complete view of system performance (specified in [D3.2]).

It should be noted that scalability should be assessed in view of the context and the criticality in which the component operates and in view of the limitations imposed by existing technologies in the network infrastructure. For this reason, the analysis is presented per distinct epoch of system operation.

3.2 RPC Epoch

Below we examine the running time cost of the Traffic Forecast Management and Network Dimensioning components, which operate at RPC epochs. Since the operations are *performed off-line*, it is not critical to keep running times very small. Rather, scaling refers to ensuring that running times grow in a polynomial fashion with respect to relevant parameters.

3.2.1 Traffic Forecasting

3.2.1.1 Scaling and Complexity of Operations

Traffic Forecasting involves the following operations (cf. [D1.2]):

Operation TFM-1: Demand Aggregation

This operation derives QoS traffic demand forecasts by aggregating the traffic of the subscribed SLSs. Traffic demand is derived perTT. The following steps are involved:

1. For each SLS calculate the maximum subscribed demand (offered load) per OA, based on the SLS contracted traffic rate(s). Assuming that in the worst case each SLS will generate traffic for all OAs, this step takes $O(|SLS| * |H|)$ time.
2. Calculate the anticipated demand per TT. For a TT, this is done by aggregating the maximum subscribed demand over all SLSs contributing to this TT derived by the first step. Aggregation is done on the basis of multiplexing factors and aggregation weights per distinguished *service class*. Service classes distinguish the SLSs based on their technical characteristics (e.g. invocation method, topological scope) according to the usage patterns observed. As such, given a certain service class, it is considered that the users of the SLSs of this class have the same SLS usage patterns; hence valid multiplexing factors could be determined. The notion of service classes is introduced to cope with the user diversity in service usage. The definition of service classes and the determination of associated multiplexing factors and aggregation weights is outside the scope of this step (and of the project in general) and may be influenced by business policies. Let SC, $|SC|$ denote the set and the number of the defined service classes. As the implementation allows direct access from TTs to their contributing SLSs, this step takes $O(|T| * |SC|)$.

In total, demand aggregation is of complexity $O(|T| * |SC| + |SLS| * |H|)$. This says that complexity is analogous to both the number of subscriptions and the number of TTs supported by the network, therefore the operation scales if these parameter scale.

Operation TFM-2: Hose Aggregation

This operation aggregates from topological perspectives the hose TTs per ingress nodes, with the purpose to restrict the total number of hose TTs in the system. This is required in order to facilitate the TE functions to feasibly engineer the network. Note that by their definition, hose TTs do not scale with the number of the edges, growing to their power.

Hose TTs can only be aggregated if they are rooted from the same ingress. When two hose TTs are aggregated, they are replaced by a new aggregated hose, which has as egresses the union of the egresses of the hoses and bandwidth the sum of the bandwidths of the hoses. Considering that the bandwidth of a hose will be delivered (engineering the network appropriately) at each of its egresses, hence there is a penalty to be paid in this aggregation procedure. The penalty relates to the difference between the bandwidth flown to the aggregated hose egresses as calculated in the aggregated hose and the bandwidth that would flow at these egresses if the two hoses were not aggregated. The maximum tolerable aggregation penalty is assumed as input.

For a given ingress node, the hose aggregation operation proceeds in an iterative fashion, aggregating hoses at each iteration as long as the aggregation penalty stays within tolerable levels. If an iteration concludes having created an acceptable number of hoses, the operation terminates. Otherwise, another iteration starts, however, with increased tolerance of the aggregation penalty.

The aggregation of two hoses requires $|V_E|$ time, and executes T_H times during an iteration, assuming that in the worst case all hose TTs will be rooted from the same ingress node, having all other network edges as egresses. The number of iterations, say I_H , depends on the initial setting of the aggregation penalty tolerance and its increment from iteration to iteration (provided as input to this operation).

Therefore, the overall complexity of the hose aggregation procedure is $I_H * T_H * |V_E|$ at the worst case.

Conclusions

Based on the previous analysis, the overall complexity of the traffic aggregation process is:

$$O(|T| * |SC| + |SLS| * |H| + I_H * T_H * |V_E|)$$

The process is scalable, provided that:

- the number of TTs in general as well as the number of hose TTs in particular as appearing in the subscriptions are scalable; these numbers are basically influenced by business policies for service provisioning,
- the number of service classes is fairly small, not spoiling the scalability of the TTs,
- the number of subscriptions is scalable, which also is influenced by business policies
- the policy for managing the tolerance of the aggregation penalty is set so that not to affect the scalability of the number of hose TTs.PP

3.2.1.2 Scaling and Complexity of Output Decision Implementation

The output of the traffic forecast process is the **Traffic Matrix**, specifying the forecasted demand per TT. It is provided as input to the Network Dimensioning component and obviously it scales as long as the number of TTs is scalable.

It should be stressed that the number of TTs included in the Traffic Matrix may be less than the TTs corresponding to the subscribed SLSs, as a result of the hose aggregation operation.

3.2.2 Network Dimensioning

3.2.2.1 Scaling and Complexity of Operations

The operations associated with the network dimensioning are the following (cf. [D1.2], [D1.3]):

1. Given a set of TTs, their associated QoS constraints and their forecasted demand, accommodate the demand in the network by determining trees and associated bandwidth for routing the TTs.
2. Determine resource availability estimates, per QoS-class between network edges. Resource availability estimates are defined in terms of minimum and maximum bandwidth parameters that can be used by each TT during system operation, on a per link and on a network-wide basis. These estimates are also used to determine various minimum and maximum bandwidth availability parameters per OA per interface, which in turn are used (by the static part of DRsM, cf. section 1.2) to configure the scheduling parameters of the PHBs.
3. Map subscribed address groups to the defined trees.
4. Determine LSPs per OA given the trees determined for QoS routing.

From a functional perspective, the last two operations are parts of the static part of DRtM (cf. section 1.2), [D1.1], [D12]. However, for the sake of this document, they are considered parts of the Network Dimensioning component, as nonetheless are executed at RPC epochs.

Operation ND-1: TT Demand Accommodation and Tree Determination

This operation is further decomposed into tasks, which in turn are decomposed into steps, as in the following:

Task 1: Pre-processing

1. Redefine in the network topology, links with large propagation delays as a concatenation of links with small delays. Therefore, this task takes at most $|E|$ time.
2. Determine the maximum hop-count K_T allowable for routes belonging to each of the hose TTs: Therefore, this task takes at most $|T|$ time.

Task 2: Optimisation Problem-Initialisation

TTs are initially allocated with trees. We start by allocating the shortest paths to each TT. The complexity the shortest path algorithm is $O(|V|^2)$, or $O(|E| * \log|V|)$ in case the implementation uses a binary heap as a priority queue. Therefore the complexity of this step is: $O(|T| * |E| * \log|V|)$

Task 3: Optimisation Problem-Execution

The optimisation algorithm is based on the Gradient Projection (GP) method [BER-GAL]. We implement the one-at-a-time mode, whereby TTs are processed sequentially and following the processing of one TT the link loads are adjusted to reflect the yielded results before carrying out the iterations for the other TTs. This is opposed to the all-at-once mode, which has worse convergence results [BER-GAL].

The algorithm (cf. [D1.2], section 4.1.3.3.4) terminates when the difference in the function values computed in two successive steps is smaller than certain percentage. For the problem at hand, the computation of optimal trees for the TTs is approximate since the problem is NP-complete. Hence convergence is not guaranteed a priori and therefore, an additional condition is placed that the algorithm stops after a certain number of iterations. Note that with this additional constraint, while the algorithm may not provide the optimal value, it can still be guaranteed that the value of the optimisation function will be better than the one provided by the initial bandwidth allocation. Assume therefore that in the worst case, \bar{C} iterations of the algorithm are performed (grossly speaking, \bar{C} corresponds to the number of alternative routes per TT).

The computations performed at each iteration are the following (cf. [D1.2], p. 78).

For each TT, that is $|T|$ times,

1. Compute link weights, based on which the TT trees (see next step) will be determined. As a TT belongs to a QoS-class, equivalently corresponds to an Ordered Aggregate (OA), each link is assigned a different weight per OA and therefore, this step takes $O(|E|*|H|)$ time. The weight assignment computation takes $O(I)$ time, as the implementation allows direct access from a link to the found trees passing through it.
2. Find a maximum hop-count constrained (cf. K_T above) minimal weight tree for each TT, according to the link weights determined in the previous step. This operation depends on the algorithm employed for the computation of constrained minimal weight trees. Approximate algorithms are employed here due to the NP-completeness of the problem. For the algorithms proposed for this problem, there is a compromise between running time and the quality (in terms of the value of the tree cost) of the obtained solution. The computational cost of the algorithm employed is a function of the network size in terms of number of nodes $|V|$ and number of edges $|E|$; let us call it $g(|V|, |E|)$. Appendix A provides an analysis of this complexity based on literature, while it also analyses the complexity of the algorithm specified by TEQUILA.
3. For each TT compute the new bandwidths of (TT loads across) its trees in $|S_T|$, redistributing bandwidth from already found trees (with positive bandwidth) to the newly found tree. Assuming that the algorithm starts with one tree per TT, the maximum number of trees per TT at any step is \bar{C} (in the worst case). For each tree it is required to compute its first and second derivative lengths [D1.3]. This computation takes $O(|E| * \log|E|)$ time, as in the worst case a tree can have as many as the network links. Therefore, this step requires $O(\bar{C} * |E| * \log|E|)$ time.

Finally, each iteration concludes with the computation of the new value for the total network cost, according to the newly calculated TT tree bandwidths (load distributions). The total network cost is the sum of the link costs, which are a function of the total loads per OA supported by the link, where the total load per OA is over all found trees using the link. Therefore, this step requires $O(|E|)$ time.

In total, each iteration requires $O(|T| * (|E| * |H| + g(|V|, |E|) + \bar{C} * |E| * \log|E|)) + O(|E|)$ time, so its complexity is $O(|T| * (g(|V|, |E|) + \bar{C} * |E| * \log|E|))$. Since we will have at most \bar{C} iterations, the total complexity of the optimisation execution task becomes $O(|T| * \bar{C} * (g(|V|, |E|) + \bar{C} * |E| * \log|E|))$. Assuming that the complexity of $g(|V|, |E|)$ is far greater than $O(\bar{C} * |E| * \log|E|)$, the overall complexity of the optimisation execution task is $O(|T| * \bar{C} * g(|V|, |E|))$.

Operation ND-2: Resource Availability Estimates

Task 1: Determine Point-to-Point Paths

Resource availability estimates are calculated per QoS-class between network edges. This task is to create all point-to-point paths from the TT trees produced by the previous operation. Note, that the produced trees correspond to the TTs against which the network has been dimensioned, which may be of either pipe or hose topological scope. The point-to-point paths produced by this task correspond to pipe TTs only.

In the context of this operation, the term TT will denote a pipe TT.

It takes $O(|V|)$ time to determine the unique path from an ingress node to one of the egress nodes of a tree, as in the worst case is taken that a path spans all network nodes. Hence, given a tree, it takes $O(|V|^2)$ to determine the unique path to all egress nodes of the tree, assuming that in the worst case all network nodes are leaves of the tree. Hence, the total running time to find the paths of all the trees of a given TT is $O(|S_T| * |V|^2)$.

In total, therefore, the complexity of this task is $O(|T| * |S_T| * |V|^2)$, assuming that in the worst case all TTs will be hoses. Considering that $|S_T|$ is at most \bar{C} (cf. operation 1), the complexity becomes $O(|T| * \bar{C} * |V|^2)$ at the worst case, and a complexity of $O(\bar{C} * |V| * (|T_P| + |T_H| * |V_E|))$ would be fairly reasonable.

Task 2: First Phase – Maximum Resource Availability

1. Determine the free capacity $R(l)$ on each link l , given the optimum load distribution of the TTs as determined by operation 1. This step takes at most $|E|$ time.
2. For each TT, which are of pipe scope in the context of this operation, consider the directed network comprised of the paths found in the previous task and determine the maximum flow (minimum cut) of this network. This is an iterative procedure, iterating for all TTs and for all their found paths. The set of paths per pipe TT consists of the paths of the TT itself, S_T , plus the paths of the hose TTs which include this pipe TT as a leg, which, following the worst-case reasoning, are $|V| * S_T$. Hence, the number of iterations is $O(|T_P| * |S_T| * |V|)$ in the worst case. At each iteration, the capacities of the links of the TT paths are revised accordingly [D1.3], which takes $O(|E|)$ time, assuming that in the worst case the set of links of all TT paths is the set of network links. Therefore, the running time of this step is $O(|T_P| * |S_T| * |V| * |E|)$.

Task 3: Second Phase – Minimum Resource Availability at Congestion

This task consists of defining appropriate parameters for each link and each TT [D1.3] for resolving contention in the links that cannot accommodate the maximum TT flows determined in the previous task; this takes $O(|T_P| * |E|)$ time. Then, the algorithm of the previous task, with the new link parameters, is employed, to determine the TT minimum flows at times of congestion. Hence, this step also takes $O(|T_P| * |S_T| * |V| * |E|)$.

In total, the computation of the resource availability estimates takes time $O(|T| * \bar{C} * |V| * (|E| + |V|))$; note that $|S_T|$ is at most \bar{C} (cf. operation 1). A fairly reasonable estimate could be $O(\bar{C} * |V| * (|T_P| * |E| + |T_H| * |V|))$.

Operation ND-3: Traffic Mapping on Trees

This algorithm [D1.2] maps address groups as specified in the subscribed SLSs to the derived trees. The input to the algorithm is, for a given TT, the set of address groups C_T from all SLSs that contribute to this TT and the TT contracted rate per SLS; this information is provided by the SLS Mgt system (cf. SSM, section 1.2). Furthermore, this algorithm is provided with the set of trees S_T and their bandwidths per TT (determined by the previous operations).

The algorithm iterates for each SLS address group, hence $|C_T|$ iterations, and at each iteration assigns the address group to a tree in S_T . It takes $O(|C_T| * |S_T|)$ time to find which address group per SLS to assign to which tree, as it searches for the unmapped address group with the maximal contracted rate and for the tree with the maximal unallocated bandwidth as yielded by the accommodation of address groups during the previous iterations.

Therefore, the running time of the algorithm is $O(|T| * |S_T| * |C_T|^2)$, which considering that $|S_T|$ is at most \bar{C} (cf. operation 1), becomes $O(|T| * \bar{C} * |C_T|^2)$.

Operation ND-4: Determine LSPs

Based on the trees the corresponding point-to-point paths between network edges per QoS-class, determined by previous operations (cf. operations 1, 2), this operation is straightforward, with complexity $O(I)$.

Conclusions

Based on the previous analysis, the overall complexity of the network dimensioning process is:

$$O\left(|T| * \bar{C} * g(|V|, |E|) + |T| * \bar{C} * |V| * (|E| + |V|) + (|T|) * \bar{C} * |C_T|^2\right)$$

The process is scalable, provided that:

- the number of TTs specified in the traffic matrix is scalable; note that this number may not be the total number of TTs as per subscriptions, as the traffic forecasting process has limited their number to desired levels. It should also be said that the number of TTs is primarily influenced by business policies for service provisioning.
- the number of address groups given away to customers is scalable,
- the number of iterations of the optimisation algorithm is kept fairly small,
- an efficient algorithm for determining hop-count constrained optimal weight trees is implemented.

3.2.2.2 Scaling and Complexity of Output Decision Implementation

The output of the network dimensioning process consists of:

- **Resource availability parameters** (given as input to SSM). These parameters specify the availability of the network per TT. As the availability parameters are fixed, this output scales as TT scale. Note that in this context, the TTs are only of pipe topological scope, not hoses, which grow polynomially with the number of network edges.
- **PHB configuration parameters** (downloaded to DRsM). These parameters refer to bandwidth and QoS requirements of each of the OAs defined on the links. The number of OAs and the number of parameters per OA are fixed. Therefore, this output is scalable, growing linearly with the number of links.
- **Trees per TT** (internally exchanged, between the process of network dimensioning). The number of computed trees per TT is at most \bar{C} , therefore this output scales, if \bar{C} is set fairly reasonably. Note that this parameter has to be in accordance with the physical topology of the network itself.
- **LSPs**: LSPs are created edge-to-edge and per QoS-class. Their number is at most $O(|T| * \bar{C} * |V_E|)$ at the very worst case; $O(\bar{C} * (|T_P| + |T_H| * |V_E|))$ would also be a fairly reasonable bound. Note that if routes that have common paths are associated to the same LSP, then the average number of LSPs created can be *significantly smaller* than this worst case. In any case, the number of LSPs is scalable as long as T is scalable.
- **LSP routing entries per network router**: At most $O(|T| * |C_T|)$ entries are created, assuming that SLSs have been defined for each TT. Therefore, routing table entries are scalable as long as C_T is scalable. It should be noted that this parameter reflects on the number of customer/service networks directly reached by a router and on the SP policy to give away IP addresses. As such, this scalability concern is a general concern of the SP operation itself, not caused by the particular design of the system.

3.3 Subscription Epochs

Below we examine the cost of operation of the Service Subscription Management component, which upon receipt of a subscription from a customer, is called to determine whether to accept the subscription, conducting relevant negotiations with the customer. Note that depending on current network availability status, a requested subscription may either be accepted as is, or alternatives could be offered or rejected.

It is generally considered satisfactory if the processing of a customer subscription takes seconds to a few minutes, and multiple customer subscriptions, in the order of a few hundreds, can be handled simultaneously.

3.3.1.1 Scaling and Complexity of Operations

Service Subscription Management involves the following operations (cf. [D1.2]):

Operation SSM-1: Negotiation Server

This operation negotiates with the customers the results of the subscription logic (see below), processing the subscription requests. It implements the server-side of the Service Negotiation Protocol (SrNP), specified by TEQUILA to enable automation of the subscription negotiation process. The operation has been designed to handle multiple customer subscriptions at the same time. Given a customer subscription session, the operation handles SrNP protocol messages received from the customer, bearing subscription requests, and the results of the subscription logic in response to the received requests. The SrNP protocol messages are stripped-off the protocol headers and are immediately forwarded to the subscription logic, and the results of the subscription logic are formed into appropriate SrNP messages and transmitted to the customer. As SrNP is dialogue-like, the operation can either be in one of the following states: waiting for an SrNP message from the customer, or waiting for the subscription logic to conclude and produce its response. Hence for a specific session the complexity of the operation is $O(I)$.

Therefore, the overall complexity of this operation is $O(|SSS^a|)$ where $|SSS^a|$ is the number of simultaneous active sessions (customer subscription requests in progress). This complexity primarily depends on implementation and the processing capabilities of the infrastructure.

Operation SSM-2: Subscription Logic

This operation processes the received subscription requests, applying the admission logic to determine whether to accept or counter-offer or reject a requested subscription. The subscription admission logic [D1.3] accepts subscriptions as long as the network could satisfactorily accommodate total anticipated RPC traffic demand per TT; otherwise negotiations are initiated. RPC traffic demand is yielded by aggregating the SLSs accepted within the current RPC including the newly requested subscription. Satisfactory accommodation is determined by checking whether RPC traffic demand per TT fits in that portion of the TT availability buffer determined to be given away to new subscriptions. The TT availability buffer is designated by the availability estimates per TT (produced by the TE functions at the beginning of the current RPC, cf. section 1.1.2); from subscription perspectives this buffer is shared between subscriptions to be requested and already established subscriptions. The portion of the resource availability buffer to be given away to new subscriptions is determined on the basis of a parameter set by business objectives, which reflects the degree of satisfaction that the SP would opt to provide to its subscriptions.

For consistency reasons, the subscription logic processes one subscription request after the other, and involves the following tasks.

1. *Analyse and translate the subscription.* A requested subscription, complying with the SSS template specified by TEQUILA, is analysed into its constituent SLSs. Each SLS is translated from customer to network parlance, mapping customer sites to network edges, quality levels as understood by customers to OAs and customer flow address groups to extended IP address groups. Hence, this task is mainly consumed by look-up's into tables holding various customer and network service data and therefore, for a given SLS, it takes $O(\log(|V_E| * |H| * |Cust| * |SC| * |SSS| * |SLS|))$ in the worst case. The logarithmic part of the expression is to account for look-up times in tables holding customer and network service data. Note that this complexity largely depends on implementation and on the efficiency of the data storage and accessing means employed in the infrastructure.
2. *Determine total anticipated RPC traffic demand.* The traffic demand (offered load) implied by a requested subscription is first derived, based on the specification of the traffic envelope in the included SLSs. The implied demand is derived per OA for all SLSs of the subscription. Assuming in the worst case that a SLS will generate traffic for all OAs, this step takes $O(H)$ time, for an SLS. Then, the total anticipated RPC traffic demand is calculated, by aggregating the SLSs established during the RPC with the requested SLS. This aggregation is not done at the level of details that was done at RPC epochs. For the purpose of this aggregation, hose SLSs are treated as multiple pipes. As aggregation of pipe SLSs is linear (since it is based on multiplexing factors and aggregation weights, cf. section 3.2.1.1), this step takes also $O(H)$ time.

3. *Determine whether to accept the subscription or call for negotiations.* This step involves a comparison of the total anticipated RPC traffic demand per TT with a pre-determined threshold specifying the maximum available resources to be given away to new subscriptions (cf. above). Hence, for a SLS, this step takes $O(|H| * \log(|V_E|^2 * |H|))$ time in the worst case. The logarithmic term is to account for the look-up required to find the predetermined threshold per TT; the size of the data to look-up is in the order of the size of the pipe TTs, which is $O(|V_E|^2 * |H|)$.

In total, assuming that in the worst case a SSS will include as many SLSs as all the ingress-egress pairs, that is $O(|V_E|^2)$, the complexity of this operation is $O(|V_E|^2 * |H| * \log(|V_E|^2 * |H|))$.

Conclusions

Based on the previous analysis, the overall complexity of the subscription process is:

$$O(|V_E|^2 * |H| * \log(|V_E|^2 * |H|)) + O(\log(|V_E| * |H| * |Cust| * |SC| * |SSS| * |SLS|)) + O(|SSS^a|)$$

The above indicates that the subscription process is scalable.

The logarithmic terms depict look-up times, which largely depend on implementation and the information processing capabilities of the infrastructure. Therefore, caution should be taken in implementation and deployment.

The number of simultaneous negotiation sessions that can be handled efficiently depends entirely on implementation and the processing capabilities of the infrastructure.

3.3.1.2 Scaling and Complexity of Output Decision Implementation

The output of the subscription process consists of:

- **SrNP protocol messages.** SrNP is session-oriented, providing means to either negotiating parties to terminate the session. Hence, the messages per session can be bound, therefore scalable. It should be noted that the dialogue-based nature of the protocol also contributes to the scaling of messages exchanged in a session. SrNP messages obviously grow linearly with the number of active negotiation sessions, which is bound by the processing capabilities of the infrastructure. Therefore, SrNP scales.
- **Subscriptions and service activation notifications.** Once subscriptions are established, they are stored in appropriate repositories. Furthermore, notifications to the appropriate components of the system (RSIM, DRtM) to undertake the necessary configurations at the routers required for activating the service for the customer are sent. Obviously, this output scales as the number of subscriptions scales.

3.4 Invocation Epochs

At service invocation epochs, the admission control function operates (part of the RSIM component, cf. section 1.2). This function is mainly required for ensuring compliance of customers to subscription profiles (hence, e.g. avoiding denial of service attacks) and non-violation of the current network state with respect to QoS delivery.

Service invocation epochs occur when TEQUILA SLS-based services are explicitly requested at the network edges (through appropriate signalling protocols e.g. RSVP) and when subscriptions for permanent (implicitly invoked) services (e.g. VPNs) are accepted. In the latter case, appropriate notification is sent by the SSM component.

On explicit service invocation, the admission control function performs authentication and authorisation. Authorisation is both administrative- and resource-based. Administrative-based authorisation checks that service invocation is in compliance to the corresponding subscription profile. Resource-based authorisation assesses that appropriate resources exist in the network to accommodate the requested QoS traffic.

In addition, the admission control function configures traffic conditioners (classifiers, policers and markers) according to the traffic envelope characteristics of the admitted services.

Service invocation admission control for explicitly invoked services is time-critical; its response times should be in the order of seconds.

As outlined in section 1.1.3, invocation admission utilises only local information, which is readily available at the time of operation. Specifically, admission control (for explicitly invoked services) implements a probabilistic admission control scheme [D1.3]. The related acceptance probabilities are set by the dynamic admission management function (see next section) according to actual network state.

3.4.1.1 Scaling and Complexity of Operations

Service invocation admission control involves the following operations (cf. [D1.3]) which can be performed independently for each service invocation:

Operation RSIM-1-1: Authentication and Administrative-based Authorisation

For an explicitly invoked service, this operation checks compliance with the subscription profile. The corresponding subscription is found first, if is not conveyed in the protocol itself. If found, checks are performed to verify the validity of the invoker/user, current time, characteristics of requested traffic envelope and topological scope against subscribed data and the already active services in the context of the subscription. Therefore, this operation is mainly consumed by look-up's into various tables holding subscription data, which take $O(DB^{ac} + \log|SSS| + |V_E|)$ assuming that an invoked service will include SLSs for all network edges in the worst case. The term DB^{ac} denotes the delay in accessing a remote database where subscription information is kept; note that invocation admission logic resides at the network edges.

Operation RSIM-1-2: Resource-based Authentication

This operation implements a RED-like admission control scheme per TT, where all related parameters are readily available, hence its complexity is $O(I)$. As the invocation may include a number of SLSs, as many as the edges in the worst case, and each SLS may generate traffic for a number of OAs, the complexity of this operation is $O(\log(|V_E| * |H|))$.

Operation RSIM-1-3: Admission Control at the B-side

When a requested service is successfully authenticated and authorised at the ingress edge, the service request is forwarded to the involved egress edges, where admission control is then performed on the physical capacity of the access interfaces. The complexity of this control is $O(I)$, as output interface information is directly accessed. Hence, the complexity of this operation is the round trip delay for conveying the invocation from an ingress edge to the involved egresses, call it DL^{e-e} . Therefore, this operation takes $|V_E| * DL^{e-e}$ time. Note that this running time depends primarily on implementation.

Conclusions

Based on the previous analysis, the overall complexity of the invocation admission control process is

$$O(DB^{ac} + \log|SSS| + |V_E|) + O(\log(|V_E| * |H|)) \text{ per service invocation.}$$

The above indicates that the invocation admission process is scalable according to its time-critical requirements, as long as it is efficiently implemented and deployed.

The number of simultaneous service invocations that can be efficiently handled by the admission control entirely depends on implementation and the processing capabilities of the infrastructure.

3.4.1.2 Scaling and Complexity of Output Decision Implementation

The output of the subscription process consists of:

- **Set of traffic conditioners** for ensuring compliance of customers' offered load to contractual levels. The number of traffic conditioners set in the edges throughout the network is $O(|SLS|)$. Therefore, this output scales as long as the number of subscribed SLSs is scalable. Note that the set of traffic conditioners is nonetheless limited by the vendor-specific capabilities of the edge routers; this very fact poses requirements on the number of SLSs and therefore on the number of subscriptions that can be established by the network.

- **Network invocation protocol messages.** A protocol is employed in the network to allow for the admission of SLSs at both end-points of the SLS. This protocol is also required for the invocation of bi-directional services. The protocol specifies two messages, one at each direction between the ingress and egress of the SLS. Hence, the protocol messages scale in a single SLS invocation. And, overall, the protocol messages scale (and overhead is tolerable) as long as the number of SLSs that can be invoked simultaneously from all network edges, call it $|SLS^a|$, is scalable; this is $O(|SLS|)$ in the worst case.

3.5 Dynamic Epochs

The running time cost and overhead of the system components operating at dynamic epochs is examined. These epochs occur on significant network load condition changes and on network state changes. Note that the latter epochs may be caused by the system itself at RPC epochs when new TE information is configured in the routers. The dynamic components of the system are: RSIM, DRsM and DRtM (cf. section 1.2).

Operations are indeed time-critical at these epochs, as the appropriate corrective actions need to be taken as soon as possible to rectify currently deteriorating network performance.

3.5.1 RSIM-Dynamic Admission Management

3.5.1.1 Scaling and Complexity of Operations

As outlined in section 1.1.3, invocation admission logic incorporates dynamic admission management functionality with the purpose to take appropriate actions to affect future and existing admissions so as to ensure that the network is not overwhelmed with traffic beyond its current capacity to deliver QoS (as per QoS-classes).

The actions taken are: setting of acceptance percentage for explicitly invoked services, which are passed to the associated admission control part of the RSIM (cf. section 3.4), and the configuration of the traffic conditioners already established at invocation epochs, with new rate limits. All these actions apply at the granularity of groups of SSS/SLSs (e.g. per customer type). For scalability and stability reasons the actions are pre-determined and ordered in severity levels in terms of their rate limits and groups of services to apply.

For scalability and stability reasons it relies on locally obtained measurements (current TT traffic) and on two network state on-off indicators (network green and network red), rather than on network load measurements. It also utilises network availability estimates per TT produced by TE functions at the beginning of the current RPC, as guidelines.

As dynamic admission management may affect active services, the frequency of taking actions is of concern. Action taking periods are considered acceptable if they are in the order of minutes and above. To this end, operation is step-sized accordingly.

Dynamic admission management operates on the TTs having as ingress the edge the admission logic (RSIM) resides, and involves the following operations (cf. [D1.3]):

Operation RSIM-2-1: Determine the Action to Apply

This operation is triggered by threshold crossing events set on the current TT traffic and on network-state changes. These are supplied by the services of the Monitoring service. The following steps are then taken:

1. *Determine the TTs affected by the received alarm.* As network-state alarms refer to a QoS-class and threshold crossing events directly correspond to TTs, the affected TTs can be directly accessed. Hence this step takes $O(1)$ time.
2. *Determine the action to apply.* For a given affected TT, the action (if any) is determined given the current network state, the pro-activeness level with which invocation logic has been configured (set by policies) to operate, and the relation of the current TT traffic with the TT availability estimates of the engineered network. This step involves basic check operations on locally available information and hence takes $O(1)$ time. Note that should an action is determined to apply, an adjacent (as in the ordered action list) to the last taken action is taken.

Conclusions

Based on the previous analysis, the complexity of the dynamic admission management function is in the order of the number of the local TTs affected by a trigger. Assuming that in the worst case all local TTs will be affected, the complexity is $O(V|_E * |H|)$.

The above indicates that dynamic admission management scales very well in terms of its decision making complexity.

3.5.1.2 Scaling and Complexity of Output Implementation

The output of the dynamic admission function consists of:

- **Configuration of traffic conditioners and probabilistic admission control schemes.** This output is analysed in terms of the frequency and the number of these configuration actions. As active services may be affected by these actions, the frequency of taking the actions is of concern. Action taking periods are considered acceptable if they are in the order of minutes and above. To this end, the operation of the dynamic admission management is step-sized accordingly, hence actions will be taken at least every such step-size. The complexity of action undertaking should be seen along the respective capabilities of the routers. As actions apply to groups of SLSs, the complexity of their undertaking is proportional to the granularity of the groups of SLSs to apply. However, although finer granularity in applying actions seems more scalable compared to coarser granularities, it is not indeed; as a matter of fact, it could lead to the undertaking of far more actions than those taken at coarser granularities at certain epochs e.g. when the network become red from green. The granularity in applying actions is input to the dynamic admission management algorithm and is considered as a topic of further investigation.
- **Monitoring requirements.** Dynamic admission management requires the load measurements of local TT traffic and the two coarse indications on network ability to deliver QoS (as per QoS-classes) given its current load. The first measurements can be performed locally through passive monitoring. The monitoring jobs to be initiated are $O(V|_E * |H| * \bar{C})$. If the second type of measurements was to be provided through active monitoring, since the monitoring jobs to be initiated throughout the network is $O(|SLS|)$, the monitoring requirements would prohibitively grow. Therefore, aggregated hop-by-hop monitoring means, as already designed by TEQUILA, need to be provided. Note that in this case, the inherent error in the measurements is not critical for the operation of the dynamic admission management. It should also be noted that the required monitoring jobs could be combined with the monitoring jobs required by the dynamic TE components (see next sections).

3.5.2 Dynamic Route Management

3.5.2.1 Scaling and Complexity of Operations

As outlined in section 1.2, DRtM, residing at network edges, is responsible for managing the assignment of the address groups as specified in the SLSs, to LSPs defined to accommodate their traffic by the network dimensioning process at the beginning of the current RPC. Reassignment is done with the purpose of exploiting the benefits of alternate routing, should more than one LSPs have been defined to accommodate the traffic of a given TT and therefore of the traffic of the contributing SLSs.

Dynamic route management epochs occur when the performance of the routed traffic of some LSPs is deemed to deteriorate. Note that for the purpose of the analysis we assume that LSPs carry traffic from only one OA. Specifically, considering an instance of a DRtM at a network edge and the set of LSPs defined by TE (at the beginning of the current cycle) for routing the traffic of the TTs originating from this edge, such epochs are triggered by threshold crossing events on the following measurements:

- PHBs QoS performance along the defined LSPs
- LSP QoS performance

The Monitoring Service provides these events. DRtM involves the following operations [D1.2].

Operation DRtM-1: PHB QoS Performance Deterioration

This operation involves the following steps.

- *Determine a subset of the LSPs that are using the PHB under consideration.* This takes $O(1)$ time in an implementation where each PHB holds a linked list containing the LSPs that are using it.
- *Determine a subset of the address groups as per SLSs, which use the affected LSPs determined in the previous step.* These steps also take $O(1)$ with the appropriate implementation (which for a LSP allows direct access of associated TTs and contributing SLSs to TTs).
- *Re-map each of the address groups found in the previous step to an appropriate LSP that is not using the PHB under consideration.* This step takes $O(|V_E| * |C_T| * |S_T|)$, or $O(|V_E| * |C_T| * \bar{C})$ time in the worst case, assuming that all local TTs were affected by the problematic PHB, only one LSP per TT was affected and all other alternative LSPs of the TT do not pass through the PHB.

Conclusions

The running time of this operation is $O(|V_E| * |C_T| * \bar{C})$ which is fairly small. Since this operation is proactive and not critical, one can examine only a subset of LSPs and address groups, determined in the previous sets, and hence the running time can be made $O(1)$.

Operation DRtM-2: LSP QoS Performance Deterioration

This operation involves the following steps.

- *Get the current load of the LSPs. That could be used as alternatives of the affected LSP.* This operation takes time $O(|S_T|)$.
- *Get the current load of the address groups currently mapped to the affected LSP.* This operation takes time $O(|V_E| * |C_T|)$, in the worst case, assuming that all possible TTs (of the OA associated with the affected LSP) where routed through the affected LSP.
- *Re-assign the found address groups, which are not currently active, to the non-affected LSPs.* This is an iterative procedure performed as many times as the found address groups from the previous step. Each iteration assigns an address group to the LSP with the maximum unallocated bandwidth. This is determined by the network availability estimates produced at the beginning of the cycle, its current load and the on-going association (from previous steps) of address groups to LSPs; hence, it takes time $O(|S_T|)$, or $O(\bar{C})$.

Conclusions

The running time of this operation is $O(|V_E| * |C_T| * \bar{C})$ which is fairly small.

3.5.2.2 Scaling and Complexity of Output Implementation

The output of the dynamic route management operations consists of:

- **Configuration of routing tables in the routers.** The number of routing tables to be configured in the network is $O(|T| * |C_T|)$ in total, at the worst case. Routing tables are configured per invocation epoch of DRtM, the granularity of which depends on actual network-state and load conditions, which to a certain extend DRtM itself influences. Following the previous analysis, at each epoch $O(|V_E| * |C_T| * \bar{C})$ routing tables will be configured at most. The complexity of their configuration should be seen along the respective capabilities of the routers.

- **Monitoring requirements.** Dynamic route management requires the monitoring of PHB and LSP performance and LSP load. If LSP performance will be monitored through active monitoring, the monitoring requirements would prohibitively grow. Therefore, aggregated hop-by-hop monitoring means, as already designed by TEQUILA, need to be provided. Note that in this case, the inherent error in the measurements is not critical for the operation of DRtM, reaction on LSP-performance is reactive and DRtM provides also for proactive actions. The monitoring of LSP loads scales as the monitoring jobs are performed locally at the network edges; the number of jobs per edge is within $O(|V_E| * |H| * \bar{C})$. Note that the required PHB monitoring jobs could be combined with the monitoring jobs of DRsM.

3.5.3 Dynamic Resource Management

3.5.3.1 Scaling and Complexity of Operations

As outlined in section 1.2, DRsM, residing at network routers, is responsible for managing the scheduling parameters of the defined PHBs per network interface according to actual load conditions. The purpose is to ensure that each PHB operates within desired levels as far as its service rate and/or target performance levels (e.g. losses) are concerned.

DRsM maintains an abstract view of a PHB. At any time, packets marked for a PHB are forwarded at a given rate, service rate of the PHB, which corresponds to a portion of the link capacity allocated to this PHB according to the scheduling discipline employed in the router. As PHB performance can be managed by appropriately adjusting PHB service rates, DRsM is concerned with the management of PHB service rates. This needs to be done in accordance of the employed link scheduling means for implementing the PHBs and should be in the range of the PHB configuration parameters produced by the network dimensioning process at the beginning of the current RPC.

Dynamic resource management epochs occur when the performance of a PHB is deemed to deteriorate and/or actual load conditions have crossed certain thresholds (determined by DRsM). Specifically, considering an instance of a DRsM at a network router and a specific interface, such epochs are triggered by threshold crossing events, provided by the Monitoring Service, on PHB performance measurements and current loads.

DRtM involves the following operations [D1.2].

Operation DRsM-1: PHB Performance Alarms

Considering a specific DRsM instance at a specific node, this operation involves the following steps:

- *Get the current load of the other PHBs, defined in the same interface with the one for which the alarm was received.* This operation takes time $O(|H|)$.
- *Reallocate PHB service rates.* Given the current loads, the previous PHB service rate allocations and the PHB configuration parameters from network dimensioning, new service rates for the PHBs defined on interface are calculated, taking into account the particular implementation of the PHB in the router. Furthermore, based on these new service rates, new threshold values are determined if load-based thresholds are used. The newly derived service rates are then configured in the routers by setting accordingly the appropriate link scheduling parameters. This step is iterative on the number of PHBs defined per interface and each iteration takes $O(I)$ as all data can be directly accessed.

Conclusions

The running time of this operation is $O(|H|)$, which indicates that DRsM scales very well.

3.5.3.2 Scaling and Complexity of Output Implementation

- **Configuration of link scheduling parameters in the routers.** The link scheduling parameters are fixed per PHB supported. Therefore, this output scales very well, however it should be seen along the respective capabilities of the routers. Such configurations occur per invocation epoch of DRsM, the granularity of which depends on actual network-state and load conditions, which to a certain extend DRsM itself influences.

- **Monitoring requirements.** Dynamic resource management requires the monitoring of PHBs in terms of their performance (e.g. losses) or current load. The required monitoring scales linearly with the number of interfaces per router and OAs supported, as the monitoring jobs are performed locally at the network nodes. Note that the required PHB monitoring jobs could be combined with the monitoring jobs of DRtM.

3.6 Monitoring System

A scalable and easily configurable architecture for performing a wide range of monitoring tasks is required. Monitoring large-scale traffic engineered networks requires mechanisms for data collection, data aggregation, data analysis, and providing feedback results. In addition, a diverse variety of measurement data is needed in order to perform network and customer service performance and traffic monitoring. The amount of measurement data will be increased in QoS-enabled networks where a number of nodes (and queues per node) and a large numbers of routes providing different QoS level service need to be monitored. Hence, the monitoring architecture must be able to scale with the size (in terms of number of routers and importance of the mesh) and the speed (in terms of bandwidth) of the network as it evolves. In order to have a scalable solution for such architecture, we propose the following approaches:

Defining the monitoring process granularity

In a DiffServ environment, the measurement methodology must be aware of different service types. Traffic engineered networks rely upon the use of classical IP routing protocols for the establishment of IP routes (shortest paths), as well as the use of the Multi-Protocol Label Switching (MPLS) technique, for the establishment of Label Switch Paths (LSP) that are expected to comply with the QoS requirements specified by the customers. IP routes/explicit paths allow control over routing of traffic flows requiring specific QoS within a domain. IP engineered routes/LSP tunnels are used to carry aggregate user traffic belonging to several SLSs having similar performance requirements. In addition, traffic engineering algorithms do not need to operate at small scale of individual packets as collecting packet-level micro-flow related statistics would be prohibitively expensive and unnecessary. Instead, observation must be performed over all packets but statistics are gathered at the aggregated macro-flow level. Hence, the monitoring process functions based on the configured classes of service handling of the data streams and the scope of offered services between ingress and egress points. That is, the measurement methodology functions at the level of PHBs and traffic-engineered IP routes/LSPs for data gathering.

Distributing the data collection system

To support the dynamic operation of the network, the monitoring architecture must be able to capture the operational status of the network without degrading network performance and without generating a large amount of data. The variety of data, the magnitude of the raw data and the necessary processing close to the measurement source make a distributed data collection system more practical i.e., one monitoring node per network element. The distributed monitoring nodes must have low impact on the performance of any router involved in monitoring and must have minimal effect on network bandwidth.

Minimising the measurement transmission overhead by processing the raw data close to the source

Processing and aggregating the raw data into accurate and reliable statistics and reducing the amount of data near its source in order to transmit the data efficiently to the traffic engineering entities is key to automating the dynamic operation of the network. The monitoring system should provide automatic threshold detection by using notification of events as well as processed measurement information. Therefore, two forms of measurement data can be considered: event notification and statistics:

Events: Event notification method can be employed to reduce a large amount of data frequently passed from monitoring nodes to traffic engineering functional entities. The granularity of event notifications can be defined for PHBs and IP routes/LSPs. Basic raw measurement data is taken in short-time scales from variables in the measurement probes. The measurement data is compared with two previously configured thresholds (the upper mark and the normal mark). If the measurement data is found to cross the upper threshold value, the relevant functional entity is informed. Depending on the measurement time-scales, event notification might be postponed on instantaneous upper threshold crossings until successive/frequent threshold crossings are observed and realised that the problem persisted for a specified time interval. This method is to insure that transient spikes do not contribute to changes unless they occur frequently. Upon event notification on upper threshold crossing, further triggers are not delivered until the measurement data returns to normal when the relevant component is notified. Threshold detection implies real-time asynchronous notification of the events that the traffic engineering algorithms needs to react to these events.

Statistics: The measurement data can be aggregated by monitoring nodes into summarising statistics in order to have a scalable system. Summarisation is usually done by integrating the measurement data over a pre-specified period. The granularity of summarisation periods must be suitably chosen based on the requirements of the interested management functional entity. The granularity of statistics range from PHB and route level for traffic engineering functions to the aggregated flow levels for customer service monitoring. Statistics should be provided in near real-time.

Using aggregate performance measurements combined with per SLS traffic measurements

The granularity of measurements can be related to SLSs since every SLS might not need to be monitored in the same way. Ideally, an SLS belonging to a premium class might need measurement results with higher frequency. Monitoring SLSs at different levels of granularity using different sampling frequencies make the monitoring architecture far more complex. Instead, monitoring every customer SLS is scalable and feasible provided aggregate network performance measurements (e.g., delay, loss, jitter) are used combined with per SLS ingress/egress traffic measurements (e.g., throughput). As several SLSs may use a single IP route/LSP, single performance monitoring action is enough to satisfy the requirement of these SLSs. As an example, injecting test traffic from an ingress point toward an egress point on a specific route for measuring one-way delay can satisfy the requirement of multiple SLSs using this same route.

Edge-to-edge vs. Hop-by-hop measurements

The scope of measurements is an important aspect of monitoring architecture. Two approaches can be taken for performing performance measurements: edge-to-edge and hop-by-hop measurements.

Monitoring scalability could be a serious concern with MPLS-TE approach. That is, if a full mesh virtual network is in-place, an order of $O(N^2)$ unidirectional LSPs needs to be monitored where N is the number of ingress nodes. It is even worse that the $O(N^2)$ order for LSPs may not be enough because multiple routes are sometimes used for load sharing or multiple services using different LSPs offered between an ingress-egress pair. In large MPLS networks, monitoring of all LSPs edge-to-edge could affect the reliability of the monitoring system that might not scale well due to the fact of having a huge number of LSPs in the network where each ingress node might need to inject test traffic to its associated routes. In addition, it might not be able to provide timely results too. Hence, LSP monitoring is scalable and feasible if only a number of LSPs are selected for edge-to-edge measurements based on some criteria/policy decisions.

The edge-to-edge approach provides edge-to-edge measurement results. The hop-by-hop approach might overcome this scalability problem by adding the hop-by-hop results and calculating an edge-to-edge result. As multiple routes may be related to a single PHB by sharing a physical link, a single test traffic sent to quantify the behaviour of a given PHB satisfies the performance monitoring requirements of these routes using that link. This results in significant reduction of test traffic in the network. With the hop-by-hop method, the status of every individual link will be known, but inaccuracy will be introduced due to the non-synchronised individual hop-by-hop measurements and concatenating these discrete measurements to estimate per-route edge-to-edge measurement values. Depending on the type of SLS that has been subscribed by the customers, this method may be appropriate for estimating the performance measurements of routes used for low profile traffic (e.g., best-effort).

Controlling the amount of test traffic injected into the network

The amount of test traffic generated by active measurements methods will be increased in QoS-enabled networks where several PHBs per node and large numbers of routes need to be monitored. There are the following requirements for test traffic:

- The test traffic load should be small compared to the load on the connection under test. If not, then the test traffic will affect the performance and the measurement does not show the real environment.
- The sampling intervals should be small enough to study fluctuations in the performance of the network.
- As the network changes over time, the amount and type of test traffic should be configurable.
- The measurements should be randomly distributed to prevent synchronization of events as described in the IP Performance Metrics (IPPM) recommendation [RFC 2330] by using a pseudo-random Poisson sampling rate.

It should be noted that the first two requirements must be as complementary as possible. That is, smaller time intervals means more test traffic, but more test traffic means a higher load on the network. A trade-off between these two requirements needs to be made. The amount of test traffic (traffic load and packet rate) on each network link, sent to measure one-way delay and packet loss during a specified time interval, will depend on the number of IP routes/LSPs crossing the link (in the case of edge-to-edge measurements), the number PHBs attached to the link interface, the number of different test packet sizes used for route and PHB related measurements, the length of these packets, and the statistical average of sampling intervals used. Practically, the test traffic should not exceed e.g., 1% of the total bandwidth that is available in the network.

4 STABILITY ANALYSIS

4.1 Introduction

Stability analysis is concerned with the assessment of the following aspects:

- *Convergence of algorithms, protocols and distributed procedures.*
- *Control-loops* between system components interacting with each other, and between system components interacting with external systems, in our case customers/users and the network. Control loops are assessed in terms of:

Convergence of operations; operations should converge to a specific network state, if such can be specified; if not, the granularity of the component invocations should be within acceptable levels, implying a sort of steady-state.

Oscillations; considering a component in a control-loop, the actions that takes (at granularities of its invocation) should not cause oscillations in network performance nor its operational state.

Design principles and mechanisms incorporated in the TEQUILA system to tackle/cope with instability issues are also described.

4.2 Convergence of Algorithms/Protocols

4.2.1 Network Dimensioning Algorithms/Protocols

For the network dimensioning algorithms (cf. section 3.2.2) stability is relevant only in ensuring that the quality of the obtained solutions does not fluctuate as the number of iterations increases and that numerical calculation problems do not arise.

Optimisation Algorithm: With the appropriate choice of parameters, the algorithm can provide an improved output at each iteration, thus fluctuations of the quality of the output can be avoided. Moreover, the algorithm stops after \bar{C} iterations (cf. section 3.2.2), a parameter that is provided as input, possibly set by policies. Hence, an improved output, relatively to the one provided at initialisation, will be provided and this operation is stable.

Resource Availability Estimates: This algorithm, by its specification, always stops and provides a solution as it is comprised from several calls to a procedure that calculates the maximum flow in a directed graph. Hence, stability is not an issue for this algorithm.

There is no issue with the stability of the algorithms **Traffic Mapping on Trees** and **Determine LSPs**, as they are deterministic.

4.2.2 Service Admission Algorithms/Protocols

Both at **subscription** and at **invocation** epochs, where corresponding algorithms run to determine whether to accept or not a service request (cf. sections 3.3 and 3.4), the decision is deterministic given specific locally available data and data of the service request. Hence, instability is not an issue at these epochs. The same applies with the dynamic invocation management algorithm at dynamic epochs; given a current internal operation state it deterministically moves to another state, taking the decisions statically associated with the new state.

The **Service Negotiation Protocol, SrNP**, operating at subscription epochs for facilitating automated service subscription negotiations with the customers, converges always, in that it terminates in finite steps, due to its very features. SrNP itself can terminate the negotiation process, after a certain number of negotiation rounds; or, either party can terminate the process by means of the `LastProposal`, `LastRevision` (aiming at mimicking a ‘take it or leave it’ situation) or `Reject` messages that SrNP provides.

There is no issue of convergence with the **network-side service invocation protocol** for controlling the admission of a requested SLS-based service invocation at both ingress and egress points. It involves two messages deterministically sent from the ingress to egress and back from the egress to the ingress of the SLS.

4.2.3 Dynamic TE Algorithms/Protocols

The **dynamic route management** (cf. section 3.5.2) and **dynamic resource management** (cf. section 3.5.3) operations are by design stable, as their operations involve the undertaking of a set of given steps, which execute deterministically given specific input data at a specific internal state. Note also, that for the network entities that they manage guidelines have been pre-determined by network dimensioning and remain unchanged during the cycle of the operation. Furthermore, this ensures data integrity amongst the distributed instances of the dynamic management functions.

4.3 Identification of Control Loops

Control loops are identified amongst the system components, which interact with other system components (internal-system control-loops) or with external systems -the network or the customers- (external-system control-loops).

Internal system control-loops are formed between:

- Interacting system components residing in different hierarchical levels, these are
 - TE configuration control-loop*: between ND and DRtM, DRsM.
 - Service activation control-loop*: between SSM and RSIM.
- Interacting system components residing in the same hierarchical level
 - RPC control-loop*: between TFM, ND and SSM.

The *external-system control-loops* are:

Subscription control-loop: involving interactions of SSM with customers for service subscription negotiation.

Invocation control-loop: involving interactions of RSIM with user applications for service invocation.

Dynamic admission management control-loop: involving interactions of RSIM -the dynamic part of its admission logic- with the network so that to take appropriate actions to affect future and existing active services when QoS performance deterioration is detected (as per supported QoS-classes).

Dynamic route management control-loop: involving interactions of DRtM with the network edges so that to re-distribute subscribed (as per SLS) address groups to defined LSPs, through routing table configuration, upon detection of network performance deterioration on specific QoS routes of the engineered network.

Dynamic resource management control-loop: involving interactions of DRsM with the network routers so that to reconfigure PHB service rates, by appropriately setting link scheduling parameters, so that to ensure that PHBs operate within desired performance levels (e.g. losses) or service rates.

It should be noted that although not directly related, the dynamic control-loops affect each other; each control-loop affects network performance and network performance changes trigger control loops. It should also be noted that, for scalability reasons, the dynamic system components are distributed across the network acting independently; no special communication protocol has been employed.

4.4 Stability wrt. Convergence of Operations

4.4.1 Issues and Requirements

The identified control-loops are assessed in terms of the *invocation granularity* of the involved components. Invocation granularity should be at desired levels depending on time-scale and context of operation. This basically means that components should react *only when necessary*, no more, no less; that is, they should operate only when required, not over-reacting, nor remaining unconsciousness to feedback information.

The formulation of a solvable problem or an approach for determining or assessing somehow the desired levels of the invocation granularity of the components of a control-loop is almost impractical due to staggering complexity, even in the definition of the very problem. Desired invocation granularities are subject then to ‘best practices’; tuning based on observation and analysis of historical data. Therefore, the issue of convergence of operations has to be primarily studied through extensive experimentation.

Our theoretical treatment on the issue concentrates on discussing aspects of components’ behaviour in the context of the identified control-loops. This is done with the purpose to amplify the analysis of the behaviour of the components and their interactions from yet another viewpoint, for identifying improvements and specifying suitable tests. Stability assertions CANNOT be done, as they can only be backed up with detailed experimentation results in diverse cases.

4.4.2 Considerations

The **internal-system control-loops** are active only at well-defined epochs; at RPC and subscription epochs, being executed once and passing the produced results to the other involved components. Hence, they behave in a stable way.

In the context of convergence, the **subscription control-loop** should terminate the negotiation sessions in reasonable time. As already explained in section 4.2.2, this is guaranteed by the inherent capabilities of the SrNP. The **invocation control-loop** should ensure that QoS signalling sessions are terminated in reasonable time. This is primarily guaranteed by the used QoS signalling protocol.

As dynamic components are distributed acting independently, and their operation is network-load driven, related to the convergence issue is the issue of *co-ordination of action undertaking* between different instances of the same component and between instances of different components. Clear objectives for co-ordinating actions of dynamic components are difficult to be instituted.

In the context of convergence, the **dynamic invocation management control-loop** is the only control-loop, which has a clear objective. To ensure that appropriate actions are taken so that when the network enters the red state for a QoS-class (no ability to deliver QoS) exits the soonest possible. The TEQUILA system by its very approach for traffic engineering ensures convergence of operations to this objective. The analysis is presented in the next section.

The stability issues of concern to **dynamic route and resource management** control-loops relate to their sensitivity in action undertaking and the impact the decisions of one component will have on the other and collectively in network performance. DRtM and DRsM see into improving network performance, each under its own perspectives. The actions of DRtM aim at exploiting the benefits of alternative routing by appropriately redistributing address groups to defined LSPs, according to actual load conditions. The actions of DRsM aim at ensuring that PHBs are adequately configured, in terms of respective link scheduling parameters, so that to provide the treatment (delay, loss) required by the QoS characteristics of the traffic routed through them.

It should be stressed that the actions of DRtM and DRsM do not contradict what so ever. On the contrary, they complement each other to the benefits of network operation. In particular, this should be seen in the context of the two-level TE approach of the TEQUILA system. Although acting independently, dynamic components are bound with the consistent traffic engineering guidelines produced at the beginning of the RPC they operate. Dynamic admission management is given network availability estimates per TT, DRtM is given LSPs defined to route injected TT traffic and DRsM is given PHB configuration parameters (ranges of service rates) for treating appropriately to their QoS requirements the traffic routed through the LSP. The specified dynamic algorithms utilise this consistent information, as configuration guidelines, mainly for resolving congestion (see next section). Hence, their actions will lead the network in near-optimal levels in cases of congestion, as engineered at the beginning of the cycle.

Since the network dimensioning output is based on certain optimisation criteria and aggregate traffic forecasts, the load fluctuations observed during actual network operation should not deviate much from the optimal. Reallocation of SLS address groups to LSPs, operates on pre-defined sets of LSPs which can be considered small relatively to the whole set of LSPs in the network. Hence, interactions with other dynamic route management instances or dynamic resource management instances are not considered severe, and the likelihood of large and sudden load shifts to other parts of the network is not big.

The above discussion provides evidence that the operation of the dynamic system components, either individually or collectively, will not result in random, unknown network states. Instead, thanks to the two-level TE approach of the TEQUILA system, the network will be driven towards near-optimum levels, as engineered, at the worst case, at the beginning of the current RPC. However, the validity of the above discussion remains at the theoretical level. Extensive experimentation needs to be undertaken to prove this claim.

Nevertheless, to cope with convergence instabilities, the dynamic route, resource and admission management components have been designed so that:

- *Their operation is step-sized.* Each step corresponds to a period of time. As a result, whatever the invocation granularity might be, the dynamic component will take actions at least every step-size periods. An efficient way to implement step-size operations is by setting accordingly the granularity of the required monitoring jobs. Step-sized operation facilitates the tuning of components according to ‘best practices’ as far as operational convergence stability is concerned.
- *They ‘understand’ that cannot longer take any other action,* and in such cases they issue warnings to higher level functions. DRtM, DRsM issue warnings to ND when all routes for certain TTs are heavily loaded or when performance of all PHBs in an interface continuously deteriorates and physical capacity keeps fully utilised. By issuing such warnings, denoting deterioration in network performance, they cease to take actions, thus stop having any effect on the already deteriorated network performance.

Stability in the above dynamic control-loops is continuously under investigation and experimentation will help into gaining more insight into the issues, identifying this way improvements in specified functionality.

4.4.3 Dynamic Admission Management Control-Loop

As outlined previously, the issue of concern to dynamic admission management control-loop is the network red-to-green convergence; should the network lose its capability to provide the QoS characteristics of a QoS-class, due to congestion, actions need to be taken to exit this state the soonest possible. The section discusses how system operations at dynamic epochs ensure such convergence.

First, it should be pointed-out that the dynamic admission management control-loop relates the dynamic admission management component not only with the network but also with the actual user who invokes traffic in the context of subscribed services, adding another dimension of unpredictability.

The problem of network red-to-green convergence should be tackled taking into account the distributed nature of system components at the network edges and the fact that all these distributed instances act independently. They are only bound by network state information. In this scenery, therefore, it is not only the actions of a single instance that need to converge; the actions of all distributed instances need to converge. The red-to-green convergence should ideally happen the soonest possible (for not keeping the network out of business), while at the same time should be ‘smoothly’ done, so that the least possible user traffic or users suffer from QoS degradation.

The algorithm of dynamic admission management components has been designed so that to offer a solution to this problem [D1.3]. It is reminded that the basic operation of the algorithm is to determine the next action to take, given current network state, QoS traffic currently admitted and the network availability estimates of the engineered network produced by network dimensioning. The actions adjust the rate limits of the traffic conditioners per SLS and the acceptance probabilities used by the admission control function; the actions are ordered in increasing severity level.

The very design of our system contributes to achieving network red-to-green convergence. When the network is engineered at the beginning of a RPC, network availability estimates are derived as well as PHB configuration parameters (ranges of service rates). In fact, network availability estimates are calculated based on the derived PHB service rates at times of congestion. In particular, network availability per TT is the maximum flow of the directed sub-network comprised of the links of the defined LSPs, with links’ bandwidths equal to the service rate at times of congestion of the PHB corresponding to the TT. When the network is in a red state for a TT, DRsM operation ensures that the bottleneck PHBs will have been ‘locked’ to rates analogous to their service rates at times of congestion. Therefore, when the network is in red, dynamic admission management implements actions to enforce the total admitted traffic per affected TT to be in the availability levels calculated when the network was engineered. Should all edges act with the same target for all their affected TTs, it is therefore guaranteed that network congestion will be resolved.

The above discussion shows that the red-to-green convergence can be feasibly realised by our system. The engineering of the network provides concrete directions (tangible targets) to the dynamic admission management components to operate in a convergent way collectively, although acting independently.

4.5 Stability wrt. Oscillations

From the identified control-loops, oscillation issues are mainly of concern to the dynamic control-loops.

Potential oscillations pertinent to the **dynamic admission management control-loop** are identified. Actions (admission control and traffic conditioning settings, cf. 3.5.1) are ordered in severity levels and are taken sequentially; that is, when it is determined to take an action, the action is either the one after the last taken or the one before. Hence, repetitive patterns can be identified and therefore oscillations can be defined e.g. when the sequence of actions is X, Y, X, Y, X is observed.

A significant cause for *action oscillations* is the user behaviour and/or the behaviour of the IP traffic itself. As an example, consider the following case. As a result of taking an action at an edge, the admitted traffic drops dramatically. By noting this behaviour, the local dynamic admission management component decides to cancel this action and takes the action preceding the last taken. However, the drop of user traffic has been due to the behaviour of elastic traffic, not because the actual user behaviour changed (quit the services). Therefore, when elastic traffic is to start building again, the action cancellation will cause the taking of the originally taken action and so on and so forth.

Moreover, action oscillations may be caused by the actions taken by other instances of dynamic admission management and/or the actions taken by the dynamic TE components; as all these components affect network state and network state changes trigger dynamic components to take actions.

In addition to action oscillations, *network state oscillations* may also occur during the dynamic admission management control-loop. These oscillations occur when the network first exits from a red state. As an example consider the following case. The network has exited from a red state. All dynamic admission management instances will try to relax the severe actions, which had to take in order to ensure that the network will exit the red state. Depending on actual user traffic conditions at the edges, it is very likely the network to re-enter a red state. In turn, the dynamic admission management instances will increase the severity of the actions, and so on and so forth.

To cope with action oscillations, the dynamic admission management has been designed to:

- Include mechanisms for identifying action oscillations, which are based on keeping track of the history of taken actions and the time elapsed.
- Increase its step-size should action oscillations are identified.

Note that action order definition should be done taking into account action oscillations. Defined actions should not be harsh, so that to smoothly reach a stable situation where no further action would be required. The 'harshness' of the actions depends on the particular traffic environment of the SP.

To cope with network state oscillations an Aloha-based scheme has been investigated; its refinement is subject of on-going work. Experimentation is required to assess the efficiency of the specified oscillation avoidance and treatment mechanisms.

There has not been identified particular oscillation concerns for the **dynamic route and resource management** control-loops.

5 CONCLUSIONS

This document presented the scalability and stability analysis of the TEQUILA system.

The TEQUILA system is not operating in a monolithic fashion. Instead the TEQUILA functionality for SLS-based service request handling and for TE is spread in a hierarchical architecture; and, operates in different degrees of time-scale, complexity and abstraction. Specifically, the TEQUILA service and TE functions operate at the following distinct epochs:

- Resource Provisioning Cycle (RPC) epochs, where the network is engineered on the basis of forecasted QoS traffic demand,
- Subscription and invocation epochs, where respective service requests are handled to determine their admission in the network, and at
- dynamic epochs, where dynamic service admission management and TE functions operate with the purpose to manage network configuration (in terms of PHB rates and QoS routes) according to actual load conditions and network state.

The input required by the operation of the TEQUILA system is

- Network topology, physical capacity and PHB capabilities
- QoS-classes, denoting the elementary QoS-based edge-to-edge transfer capabilities of the network, based on which QoS-based services will be offered; each QoS-class corresponds to a PHB Group (cf. section 1.1.1).

scalability results

complexity of operations

Viewing the system as a collection of decision-making components, scalability was assessed both at the level of decision-making and decision-implementation, assessing therefore the storage requirements of the system results. The (worst-case) scalability analysis yields that:

- At RPC epochs, the traffic aggregation function for forecasting QoS traffic demand scales with the number of the subscribed SLSs and the elementary QoS-classes, corresponding to PHBs, supported by the network.
- At RPC epochs, the network dimensioning process scales as long as the number of entries in the traffic matrix scales, reflecting the commodities –elements of traffic anticipations- against which the network will be dimensioned and an efficient optimisation algorithm is implemented. It scales also with the number of address groups given away to customers. It should be noted that the number of iterations of the optimisation algorithm should be kept fairly small. In essence, this number determines the number of alternative QoS routes between network edges.
- At subscription epochs, subscription negotiation scales with the number of ingress-egress pairs and the number of QoS-classes supported. Caution should be given to implementation and deployment. The number of simultaneous negotiation sessions that can be handled efficiently depends entirely on implementation and the processing capabilities of the infrastructure.
- At invocation epochs, admission control scales with the number of edges and as long as it is efficiently implemented and deployed. The number of simultaneous service invocations that can be efficiently handled by the admission control entirely depends on implementation and the processing capabilities of the infrastructure.
- At dynamic epochs, the dynamic service admission management and TE functions, for managing routing and PHB rates, scale with the number of edges and the number of supported QoS-classes.

data requirements

The data generated by and exchanged internally between system functions is:

- SSS/SLS: It holds the subscribed contracts and their constituent SLSs.

- **Traffic Matrix:** It holds QoS traffic forecasts calculated by established subscriptions. It has as many entries as many entries as the distinct <ingress-egress(es), QoS-class> tuples appearing the subscribed SLSs.
- **Resource availability matrix:** It holds parameters specifying the availability of the engineered network for QoS traffic between its edges. It has as many entries as the <ingress-egress, QoS-class>.
- **PHB configuration parameters:** These parameters refer to bandwidth and QoS requirements of each of the PHBs defined on the links. It has as many entries as the number of network links, where each link counts as many times as the number of PHBs it can support.
- **Trees for QoS routes:** It holds the trees determined by network dimensioning for routing the forecasted QoS traffic demand. It has as many entries as the entries of the Traffic Matrix times the number of alternative routes found.
- **LSPs:** It holds the LSPs determined by the network dimensioning process. At the worst case it has as many entries as the previous matrix times the number of network edges.
- **LSP routing entries per network router:** It holds the mapping of address groups as per subscribed SLSs to a determined LSP. It has as many entries as the addresses given away to customers.
- **Link scheduling parameters:** It holds parameters required for the configuration of the link scheduling capabilities of the routers for implementing the supported PHBs. They are determined based on the PHB configuration parameters. It has as many entries as the table holding the PHB configuration parameters.
- **Traffic conditioners parameters:** It holds the required parameters (flow description, actions) for setting traffic conditioner for ensuring compliance of customers' offered load to contractual levels. It has as many entries as the number of subscribed SLS.

From all above data, the latter four tables are to be finally configured in the routers.

It is evident that the requirements for efficient data storage and accessing are high. Data access has been accounted in the complexity analysis presented. It should be noted that the data is not required to be kept centralised, as the TEQUILA system is distributed.

monitoring requirements

Only the components operating at dynamic epochs require monitoring. Specifically, these components require performance and load statistics on the defined LSPs and load statistics on the PHBs. The components require only threshold crossing events.

To accommodate the above monitoring requirements, monitoring jobs at each router node need to be established for each PHB; in addition, in edge routers monitoring jobs need to be established for all LSPs emanating and terminating.

scalability conclusions

The scalability analysis indicates that the system is scalable, as it scales linearly with the entities of the environment: number of subscribed SLSs, QoS-classes, nodes, links, address groups given to customers. Furthermore, the analysis has showed that the time-critically requirements of the operation of the various system functions can indeed be met.

It should be stressed that complexity decreases significantly as we move down the hierarchy of the TEQUILA system. From complexity of the order of the number of subscribed SLSs and entries of the traffic matrix, at RPC epochs, complexity is dropped to the order of ingress-egress pairs, at subscriptions, and to the order of edges, at invocations, to finally become to the order of PHBs (DRsM).

This proves the validity of the TEQUILA approach to service admission and TE, which can be summarised as 'initially plan and then take care'. This way, heavy-processing functions is taken off-line at higher levels of the system hierarchy. Lower-level functions become really light-weight, operating, with very favourable complexities, utilising the results produced by the higher-level functions. By realising this approach through a hierarchical architecture, spanning different time-scales, the complexity of functions operating real-time to network state dynamics and service requests is reduced, as complexity has been given away to the higher system functions, which operate off-line.

Finally, it should be pointed out that for the operation of the dynamic aspects of the system, requirements for scalable monitoring solutions are high. It should be noted that if such requirements cannot be met, due to its modular design, the TEQUILA system could still perform its task and engineer the network to deliver QoS (co-operating with data plane functions for routing and resource control).

stability results

The system functionality has also been assessed from stability perspectives. Control-loops were identified, and convergence and oscillations concerns were analysed. The analysis shows that:

- Specified algorithms and protocols converge
- Convergence to acceptable network states, where QoS can be provided, from non-acceptable ones is guaranteed by the operation of the service admission functions of the system and the co-operation between the service admission and TE functions
- Convergence to near-optimum network state is guaranteed by the adopted two-level ('initially plan, then take care') TE approach and the concept of RPC (resource provisioning cycle at the granularity of which the network is engineered). In essence, the dimensioning results produced by the off-line TE functions at the beginning of RPC, are used by the dynamic functions as guidelines for resolving congestion.
- Oscillations caused by unpredictable users' and/or traffic behaviour were identified and the operations. The system, thanks to its service admission functions and in co-operation with the TE functions, provides means to avoid oscillations

All the above should be treated as theoretical analysis. Extensive experimentation is required to verify their validity. Guided by the results and the insight gained from the presented analysis, tests have been undertaken to assess the validity and performance of the system; these tests and their results are presented in Part A of this Deliverable.

6 APPENDIX A – ANALYSIS OF CONSTRAINED STEINER TREE ALGORITHMS

The algorithmic complexity $g(|V|, |E|)$ of the constraint Steiner Tree algorithm required in step 2b, varies depending on the performance of the algorithm in terms of the resulted tree cost. In the simplest case one could use a shortest path algorithm, which has complexity $O(|E| * \log|V|)$, assuming the required priority queue is implemented as a binary heap [CORMEN]. Though the shortest path algorithm is a simple solution, it does not give us the best tree in terms of total cost and it does not cater for the number of hops (hop count) per tree branch as required by the Network Dimensioning algorithm.

We have defined a solution to this problem in D1.3, which is known in the literature as the constraint Steiner Tree problem. Solutions have also been investigated for the same problem in another context, that of multicast tree formation with delay constraints. The most famous algorithms for this problem are the one proposed by Kompela et al (KPP) [KPP] and Zhu et al (BSMA) [BSMA]. The BSMA gives much better trees in terms of total tree cost but at the expense of extra complexity [SALAMA]. The complexity of KPP is $O(\Delta * |V|^3)$, where Δ is the delay constraint of the tree. In order for the algorithm to be polynomial Δ must be a bounded integer, thus loosing in accuracy. BSMA uses a k -shortest path algorithm, and has complexity equal to $O(k * |V|^3 * \log|V|)$, where k is the average number of shortest paths computed by the k -shortest path algorithm, which in the case of large networks this will have great impact on the performance.

In D1.3, section 2.1 we have proposed another alternative algorithm for the constraint Steiner tree problem. Steps 1-2 require the same complexity as a shortest path algorithm, that is $O(|E| * \log|V|)$. In step 3 we use the shortest path algorithm, as many times as the number $|V_s|$ of the egress points of the tree, so the complexity is $O(|V_s| * |E| * \log|V|)$. The following steps 4-14 iterate as many times as the egress nodes, $|V_s|$, since at each iteration one of these nodes is added to the current tree. Step 4, iterates over all the pairs (Cartesian product) of the current egress nodes and the tree nodes, that is the set $V_s \times V_T$, and since the set V_T , tree vertices, has cardinality $O(|V|)$, the loop has $O(|V_s| * |V|)$ time complexity. Now we have an if condition, with the first part (steps 5-6) to have worst-case complexity as the previous step. The other part of the condition has number of steps, which apart from the trivial $O(1)$ cases the main contributors in complexity are, step 9 which runs for each egress, so its has $O(|V_s| * |E|)$ complexity, and step 12 which is the loop-breaking procedure. Step 1 of the procedure has $O(|V_T|)$, where V_T is the set of vertices of the tree, and then steps 2-10 are repeated $O(|V_T|)$ times with the while loop in steps 5-7 having complexity $O(|E_T|)$ since it might run over all the tree links, so the whole procedure has complexity: $O(|V_T|) + O(|V_T| * |E_T|)$, and since $|V_T|$ is $O(|V|)$ and $|E_T|$ is $O(|E|)$ the loop breaking procedure requires at worst-case $O(|V| * |E|)$. So the total complexity of steps 4-14 of the main algorithm is: $O(|V_s|^2 * |V|)$ or $O(|V_s| * |V| * |E|)$. If we assume that the number of egress nodes is proportional to the total number of nodes, $O(|V|)$, then the worst-case complexity of the algorithm is $O(|V|^3)$. Note that the worst-case analysis above makes assumptions like that $|V_T|$ and $|V_s|$ are $O(|V|)$ and $|E_T|$ is $O(|E|)$ which is not true in reality.

7 REFERENCES

- [D1.1] D. Goderis et al, “Functional Architecture and Top Level Design”, Deliverable D1.1, IST TEQUILA Project, 2000.
- [D1.1] D. Goderis (ed.) , D1.1: Functional Architecture Definition and Top Level Design, CEC no. 101/Alcatel/b1, July, 2001.
- [D1.2] P. Trimintzios (ed.), D1.2: Protocol and Algorithm Specification, TEQUILA Consortium Deliverable, CEC no. 102/UniS/b1, January, 2001
- [D1.3] P. Van Heuven (ed.), D1.3: D1.3: Intermediate-Results based Protocol and Algorithm Specification, TEQUILA Consortium Deliverable, CEC no. 103/IMEC/b1, October, 2001
- [D2.3] D. Griffin (ed.), D2.3: “Status report on the simulation, testbed and testtool implementations, TEQUILA Consortium Deliverable, CEC no. 203/UCL/b1, October, 2001
- [KPP] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicasting for Multimedia Applications", In Proc. of IEEE INFOCOM'92, pp 2078-2085, 1992
- [BSMA] Q. Zhu, M. Parsa, and J. Garcia-Luna-Aceves, "A Source-based Algorithm for Delay-constraint Minimum-cost Multicasting", In Proc. IEEE INFOCOM'95, pp. 377-385, 1995
- [SALAMA] H. F. Salama, D. S. Reeves, and Y. Viniotis, "Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks", IEEE J. Selected Areas Commun. (JSAC), vol. 15, pp. 332-346, April 1997
- [CORMEN] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, "Introduction to Algorithms", MIT Press, 1990
- [BER-GAL] D. Bertsekas and R. Gallager, "Data Networks", Prentice Hall, 1992