Simple Economic Management Approaches of Overlay Traffic in Heterogeneous Internet Topologies

European Seventh Framework Project FP7-2008-ICT-216259-STREP

Deliverable D3.4
Economic Traffic Management Systems Architecture Design (Final Version)

The SmoothIT Consortium

University of Zürich, UZH, Switzerland
DOCOMO Communications Laboratories Europe Gmbh, DOCOMO, Germany
Technische Universität Darmstadt, TUD, Germany
Athens University of Economics and Business - Research Center, AUEB-RC, Greece
PrimeTel Limited, PrimeTel, Cyprus
Akademia Gorniczo-Hutnicza im. Stanislawa Staszica W Krakowie, AGH, Poland
Intracom S.A. Telecom Solutions, ICOM, Greece
Julius-Maximilians Universität Würzburg, UniWue, Germany
Telefónica Investigación y Desarollo, TID, Spain

For more information on this document or the SmoothIT project, please contact:

Prof. Dr. Burkhard Stiller
Universität Zürich, CSG@IFI
Binzmühlestrasse 14
CH—8050 Zürich
Switzerland

Phone: +41 44 635 4355
Fax: +41 44 635 6809
E-mail: info-smoothit@smoothit.org

Document Control

Title: Economic Traffic Management Systems Architecture Design

Type: Public

Editor(s): Zoran Despotovic

E-mail: despotovic@docomolab-euro.com

Author(s): Fabio Hecht, Peter Racz, Burkhard Stiller, Zoran Despotovic, Christian Gross, Konstantin Pussep, Sergey Kuleshov, Marcin Niemiec, Maria Angeles Callejo Rodriguez, Sergios Soursos

Doc ID: D3.4-v1.1.doc

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
V0.1	May 25, 2010	Zoran Despotovic	First version providing relevant parts from the initial architecture design
V0.2	June 2, 2010	Zoran Despotovic	Revised table of contents
V0.3	June 28, 2010	Fabio Hecht, Peter Racz, Zoran Despotovic, Christian Gross, Konstantin Pussep, Sergey Kuleshov, Marcin Niemiec, Maria Angeles Callejo Rodriguez, Sergios Soursos	First inputs from all involved partners
V0.4	July 13, 2010	Fabio Hecht, Peter Racz, Burkhard Stiller, Zoran Despotovic, Christian Gross, Konstantin Pussep, Sergey Kuleshov, Marcin Niemiec, Maria Angeles Callejo Rodriguez, Sergios Soursos	Second round of contributions from all partners
V0.5	July 16, 2010	Fabio Hecht, Peter Racz, Zoran Despotovic, Christian Gross, Konstantin Pussep, Sergey Kuleshov, Marcin Niemiec, Maria Angeles Callejo Rodriguez, Sergios Soursos	Review ready version, inputs from all partners
V1.0	Aug 15, 2010	Fabio Hecht, Peter Racz, Zoran Despotovic, Christian Gross, Konstantin Pussep, Sergey Kuleshov, Marcin Niemiec, Maria Angeles Callejo Rodriguez, Sergios Soursos	Version with reviewers comments integrated
V1.1	Aug 24, 2010	Burkhard Stiller	Final corrections and adaptations

Table of Contents

1 Executive Summary

The goal of this deliverable is to describe the final version of the SmoothIT architecture that supports Economic Traffic Management (ETM) mechanisms in order to manage and optimize overlay application traffic in the network of Internet Service Providers (ISP) and telecommunication operators.

This deliverable is the result of tasks T3.2 “System Architecture Design”, T3.3 “Development of Implementation Architecture” and T3.4 “Implementation of Economic Traffic Management Mechanisms”. An initial version of the SmoothIT system architecture was documented in deliverable D3.1. However, the knowledge and experience gained throughout the project and especially the implementation of the economic traffic management mechanisms led us to revisit and redefine some choices made in the initial version of the architecture. These changes are documented in this deliverable. The deliverable also confirms the full realization of milestone M3.4 “System Architecture Definition Performed (Final)”.

The deliverable presents the main architectural components and their basic interactions. All involved architectural components are briefly described, along with the main interaction patterns between them. We show how our implemented ETM mechanisms fit this architecture and what specific interactions are needed among relevant components in case of our BGPLoc, IoP, and HAP mechanisms. We also revisit the design space presented in deliverable D3.1 and show how our mechanisms map to the original design options. The system requirements are another topic addressed in this deliverable. We comment on what requirements were fulfilled and how, giving thus a look back through the project and summarizing its overall contribution.

Therefore, main results of this deliverable include:

- The overview of the overall architecture: What components are in the system and how they interact in order to achieve the imposed objective (Section 4).
- Design of ETMs in the context of the introduced and defined architecture (Section 5).
- Analysis of how well the architecture meets the requirements set earlier in the project (Section 6).
- A description of the design spaces and their relation to the implemented ETM mechanisms (Section 7).

2 Introduction

This deliverable presents the final system architecture designed to enable and support an easy deployment of the economic traffic management mechanisms developed throughout the project. Besides, the architecture is designed to be able to support not only mechanisms the SmoothIT team developed but also any other mechanisms that can be eventually designed. Thus any future algorithms to manage overlay traffic based on exchange of information between overlay and underlay networks can be easily built upon the architecture we designed. The purpose of this deliverable is among others to show how this can be done.

We structured the document in such a way to make this point as clear as possible. After revisiting the used terminology in Section 3, we present in Section 4 the overall architecture, giving general descriptions of the involved components and interactions among them. Instead of giving detailed descriptions of the involved components and their detailed functionalities in the context of the overall architecture, we decided to take a bit different viewpoint and describe them by giving details of their specific operations in the context of the implemented ETM mechanisms. This is the topic of Section 5. The BGPLoc, IoP and HAP mechanisms are described in this section along with detailed descriptions of responsibilities of each involved component.

Section 7 gives a broader view on how the proposed architecture can be used in the context of design space possibilities discussed earlier in deliverable D3.1. A general view on design space options is presented and a mapping of the implemented mechanisms to design space is presented next. Besides the mechanisms we implemented, we also provide a view on combinations of ETM mechanisms.

Section 6 presents our discussion on how we addressed the requirements set in the beginning of the project and redefined later. We repeat the list of requirements we came up with in WP1 and describe how our architecture and specific ETM mechanisms fulfill those requirements. Our main conclusion is that the requirements set are almost completely fulfilled.

We finally stress that this deliverable concludes the work in WP3 and thus it presents the final report of that work package. It can be also seen as a summary of the ideas and achievements documented in the other deliverables of the WP3, most notably D3.1 (Economic Traffic Management Systems Architecture Design - Initial Version) and D3.3 (Documentation of Engineering and Implementation – Final Version).

3 Terminology

This section covers the key terms as well as its explanations utilized for the SmoothIT architecture design.

Internet Service Provider (ISP)

ISPs are commercial providers that offer Internet connectivity to users and companies. Between different ISPs, agreements are in place to regulate how traffic is routed from one ISP to another and how it is charged. Peering and transit agreements are the most common settlements for ISPs.

Peering Agreement

Peering agreements are common between two ISPs that are approximately of the same size, have similar data volumes and are geographically located in the same region. This kind of agreement is a mutual understanding to forward the traffic from the partner ISP for free. No money is being exchanged in this kind of settlement.

Transit Agreement

Due to the different sizes of ISPs, the smaller ISPs rely on the services of larger ones to connect their customers to the Internet. Transit agreements are settled on financial basis and describe a customer relationship between a larger ISP and a smaller one. Prices are based, for example, on data volume or bandwidth consumption.

Intra-domain Traffic

Traffic that stays within the borders of one ISP domain is referred to as intra-domain traffic. It does not incur additional costs for the ISP.

Inter-domain Traffic

Inter-domain traffic is the counterpart to intra-domain traffic and describes traffic that leaves the domain of one ISP. Inter-domain traffic may incur additional costs to the ISP, depending on the kind of agreement that is in place with the neighboring ISP that receives the traffic.

Overlay Networks

An overlay network is a logical network built by applications to use alternative routing mechanisms independent from IP routing and achieve an abstraction from the network, *e.g.*, new services and features not deployed in the Internet like multicast can be emulated on application layer over virtual overlay networks. Overlay applications include, but are not restricted to, peer-to-peer networks.

Peer-to-Peer Network

A peer-to-peer network is an overlay network in which peers – normally home computers – both provide and consume resources. Examples of services offered by peer-to-peer networks are file sharing, Voice-over-IP (VoIP) and video streaming.

Economic Traffic Management (ETM)

SmoothIT proposes a new traffic management mechanism termed Economic Traffic Management (ETM), which provides for incentive-compatibility in interactions between overlay applications and the underlying ISP networks in order to gain the following measurable impacts:

1. Cost saving for ISPs: lower operation costs, due to ETM-based traffic engineering, lower interconnection costs, since traffic can be kept inside an ISP's domain, and lower capacity extension cost, since capacity requirements can be forecasted with much higher accuracy.
2. Lower prices for end-users, due to competitive pricing by the ISP, which are enabled by new ETM mechanisms.
3. Better Quality-of-Service (QoS) for overlay-based applications across ISP domains, due to the usage of ETM-based traffic engineering. This leads to an improved media consumption experience for end users.

Incentive

An incentive determines a monetary or non-monetary factor which provides a motivation for a particular course of action, or counts as a reason for preferring one choice to another.

SmoothIT Information Service (SIS)

SIS is a service that conveys information between overlay application and network. It is accessed by overlay applications and provided by a network operator in order to achieve ETM of overlay application traffic.

SIS Client

An SIS Client is a client of the SmoothIT Information Service that is located in the overlay application and accesses the SIS server

SIS Server

An SIS Server is a server or a server farm that provides the SmoothIT Information Service and replies to requests from SIS clients.

4 Architecture

The SmoothIT architecture in its final version is outlined on the top-level and its components.

4.1 Top-level Architecture

Figure 1 depicts the SmoothIT high level architecture. Every ISP deploys a SIS Server that is the core of the architecture. It is responsible for providing support to the overlay formation process so that both the overlay and the ISP benefit. This must hold irrespective of whether the overlay spans multiple ISPs or the SIS Servers of multiple ISPs communicate. However, as the figure shows, we expect that all non-tier-1 ISPs will deploy their own SIS as there is a clear incentive for that and the communication between them might provide additional benefits. Note that SIS Servers can be implemented as distributed systems, although the figure shows each of them as a single box.

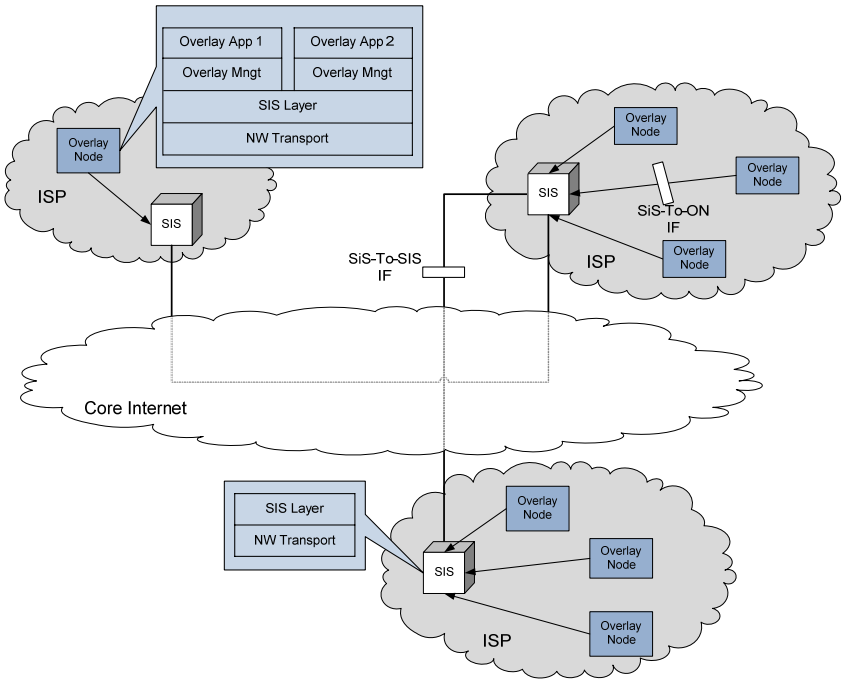

Figure 1: Top-level Entities

The figure also shows that the SIS should effectively operate as a middleware component which can be plugged into any overlay application between the overlay management layer and the network layer. Different applications normally have different QoS requirements and the SIS must be able to tune its operation according to the requirements obtained from the application.

4.2 Components

The components of the SmoothIT architecture are shown in Figure 2. The SIS Controller is a central component providing interfaces to the SIS Client and the IoP and using the services of other components, like the Metering, QoS Manager, SIS DB.

The components of the architecture are described in detail in the following sections. They communicate with each other through well defined interfaces, which allows for flexible support of various ETM mechanisms, by enabling and disabling certain components or changing their implementation.

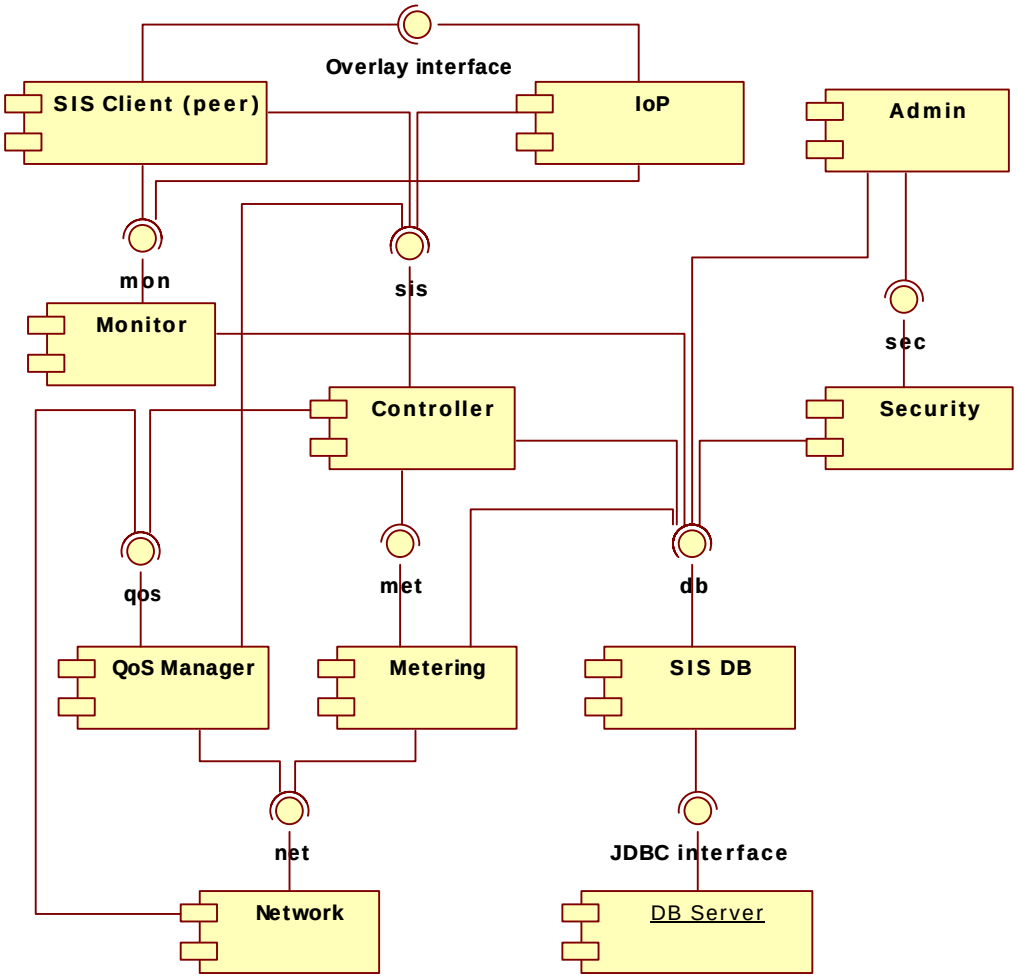

Figure 2: Components of the SmoothIT Architecture

- Swarm rating: for the IoP ETM, the Controller implements the swarm rating function in so as to advise the IoP which swarms are beneficial for it to join. It collects overlay statistics from the local peers, and after processing them, it decides on their popularity and advertises the more popular to the IoP.
- HAP rating: for the HAP ETM mechanism, the Controller decides which of the local peers in the ISP's domain are behaving well with respect to the ISP's objectives and promotes them by assigning them extra resources to serve their purposes and the rest of the (local) swarm.

4.2.1 Client (Peer)

SmoothIT peer-to-peer client is built around the NextShare P2P application developed in another EU project [p2p-next]. NextShare is a BitTorrent-based P2P file-sharing

application that additionally provides a video on demand feature. However, it is important to notice that the rest of the SmoothIT architecture is designed to be easily integrated with any P2P client, i.e., our NextShare-based client is not a requirement that we impose, but rather our choice to demonstrate feasibility of our solutions.

4.2.2 IoP

The SIS architecture involves another type of client (peer), namely the ISP-owned peer (IoP). This component is a peer created and managed by the ISP with the goal to attract and promote local traffic exchange as much as possible without any change in the used p2p protocol. The IoP receives swarm-related information from the SIS controller and decides based on this information which swarms to join to. This decision is the crucial part of the IoP operation and is documented in detail in deliverable D2.3 [D2.3]. On the other hand, The IoP communicates with other peers participating in the selected swarm and this enables retrieving more information about the swarm itself.

4.2.3 Database

The database component is the central point for storing different types of data, which is required by other SIS components. This includes the following data:

- configuration parameters for each component
- underlay data obtained by the Metering component
- statistical overlay data obtained by the Monitoring component and the Controller
- login data such as usernames and passwords

The configuration database offers a unified interface to other components, so that they can store, update, or delete their corresponding information. The configuration database makes use of the Java Persistence API (JPA) in order to be independent from the underlying database being used. During the development and during the internal and external trial, MySQL 5.1 was deployed.

In order to ease the usage of the database component the Data Access Object and Data Transfer Object (DAO/DTO) pattern was used which is build on top of the JPA. In doing so, the query functionality of the database is encapsulated transparently in POJO which can be instantiated and used by other server side components easily.

4.2.4 Metering

The metering component is responsible for collecting any information from the network that is required by ETM mechanisms and components of the SmoothIT architecture. This information is provided to other components in the form of metering data. Based on the metering data received from the metering component, the SIS can perform its task and assist overlay applications in their peer selection process. Metering data can also be used for accounting and monitoring of overlay applications.

The metering data can include information about the network, like its current state, its topology, and its performance metrics. In case of the BGP-Loc ETM mechanism [D2.3] the metering component collects BGP (Border Gateway Protocol) routing information from BGP routers of the ISP. Based on the routing information the metering provides generic rating values of IP destinations to the SIS controller. BGP is an Exterior Gateway Protocol

(EGP) used for routing information exchange between Autonomous Systems (AS). An AS is a group of networks under the common administration of a single network operator and having a common set of routing policies. ISPs implement inter-domain routing based on BGP according to their business relations with other ISPs. Therefore, the BGP routing information can be used by the SIS to assist inter-domain peer selection. Details of the BGP-Loc ETM mechanism can be found in [D2.3].

The metering component retrieves the routing table via SNMP (Simple Network Management Protocol) from one of the BGP routers of the ISP. It reads the standard BGP SNMP MIB (Management Information Base) in order to retrieve the relevant BGP attributes of each routing entry. The metering provides an interface over which the SIS controller can get the generic rating values of IP destinations and use this information to provide locality information to overlay applications.

4.2.5 Security

The security module is a component which is responsible for security assurance. It provides the basic security services which are based on the following requirements: authentication, access control, and non-repudiation. Additionally, the component is able to ensure confidentiality and data integrity. The security component consists of an AAA server (Authentication, Authorization, Accounting) as well as some additional services which are integrated with JBoss environment such as data encryption and sign of messages.

Authentication service is always required when administrator or another SIS module wants to establish a connection. In this situation, validation of the identity is obligatory. Several methods of authenticating an entity can be implemented. Currently the component supports static passwords. When i.e. the administrator wants to connect to SIS web console, he/she will send a request to access and then he/she will have to send authentication credentials: login and password. After successful authentication, administrator can connect to SIS.

Authorization service deployed in the security component creates access control mechanisms in order to ensure proper rights the entities have. After successful authentication, SIS server checks the user role. If user has appropriate role, it will have access to some resources. Otherwise it will get information that it does not have permissions.

Accounting service will be created especially for SIS. Collected information in the log-file consists at least of data about each event, username and effect of the event.

In order to activate encryption to ensure data confidentiality, some XML's files need to be configured. The key store parameters and types of certificates need to be specified. Similarly, the sign of message is ensured.

Besides basic security services the security component provides some other features, which increase security level of SIS. These improvements were not available in basic JBoss server functionality and were designed and implemented specially for SmoothIT project. Below the most important extensions are listed (the deliverable D3.3 presents these services in detail).

- Hashes of password
- No password caching

- Automatic account creation on first deployment

4.2.6 QoS Manager

The QoS Manager component is required for the implementation of the HAP mechanism. It has interfaces with the network, the Controller and the Configuration DB. It is used during the calculation of the HAPs and during their upgrade in the following way:

- The QoS Manager interfaces the network in order to retrieve the bandwidth profiles associated to the clients whose IP addresses are candidates for HAP promotion. This information is used by the Controller to calculate all those peers whose bandwidth must be upgraded. The HAPs are stored in the Configuration Database.
- The NMS requests the list of HAPs from the QoS Manager periodically. When this request arrives, the QoS Manager retrieves this information from the Database.

4.2.7 Admin

The Admin component is a web tool to administrate, monitor and fine tune the operation of the other components in the system. It offers a Web-based interface toward other components in the system, presenting a number of possibilities to retrieve or set specific parameters of those components. For example, an administrator may want to update the information on relationships with other ISPs or data related to the P2P application operation, such as maximum allowed peer request rate. Or, he or she can set the range of IP addresses to be used for locality promotion.

The Admin interface has been implemented as a set of JSP/JSF pages and as such is highly modular, i.e., it is easy to replace or change any specific functionality of the component.

4.2.8 Monitor

The Monitor component is a supportive component that collects overlay data for evaluation purposes. More precisely, it retrieves over a simple HTTP-XML interface the client statistics reports, parses the data and stores it in the DB. These data can be then used for the evaluation of the various ETM mechanisms, by providing a clear picture of the overlay activity of the local peers.

5 Design of ETMs

In this section we present the design of the three ETM mechanisms, which were selected out of those proposed in WP2 [D2.3] and have been implemented in the SmoothIT prototype. The ETM design presents the mapping of devised mechanisms into the SmoothIT architecture at the level of architectural components, their interfaces, their interactions, and even the exact data types of stored and exchanged values.

5.1 BGP-Loc

The SIS architecture components related to the BGP-Loc ETM are shown in Figure 3. The BGP-Loc ETM makes use of the metering component to rate peers and provide the rating information to the SIS clients (typically P2P clients) through the SOAP-based rating interface.

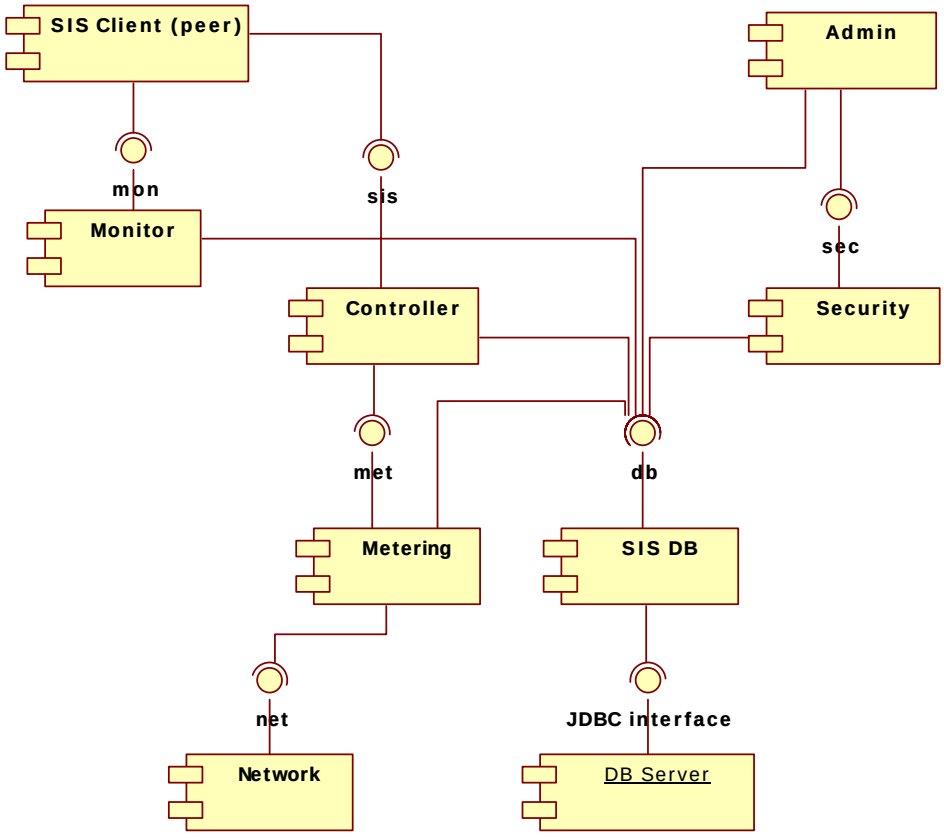

Figure 3: SIS architecture for BGP-Loc

5.1.1 Interactions

Relevant interactions cover the peer rating and the reading of BGP information.

5.1.1.1 Peer Rating

The SIS Client can receive peer ratings from the Controller over a SOAP-based interface as shown in Figure 4. The Controller uses the generic rating parameters of IP addresses

retrieved from the Metering to calculate the SIS preference value. The SIS preference value is calculated according to the BGP-Loc ETM algorithm [D2.3].

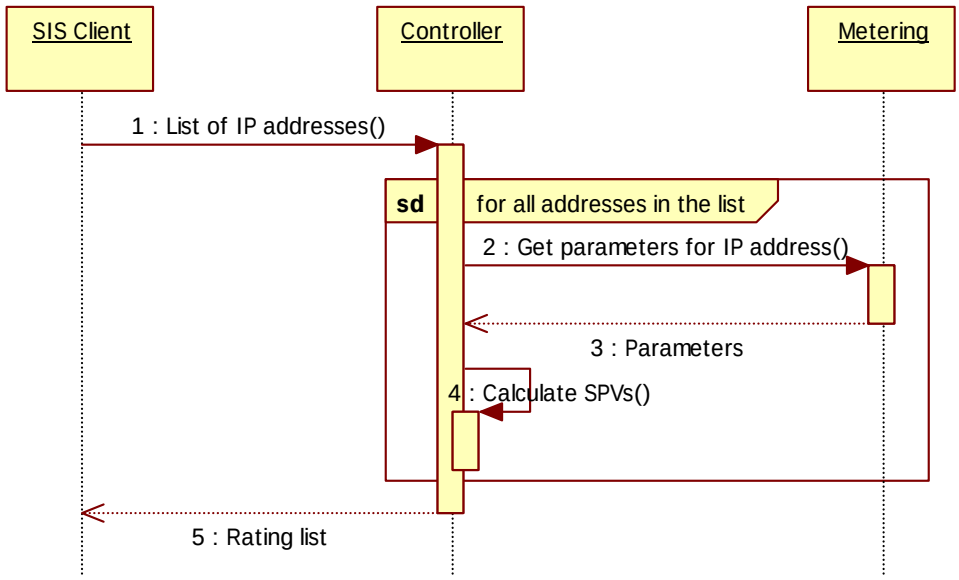

Figure 4: Peer rating

The retrieval of the SIS preference values includes the following steps (see Figure 4):

1. The Client sends the list of peers (IP addresses) to the Controller.
2. The Controller retrieves the rating parameters for each IP address in the list from the Metering.
3. The Metering returns the parameters.
4. Based on the parameters the Controller calculates the SIS preference values (SPV) for each IP address in the list.
5. The Controller returns the rated list of peers to the Client. The Client contacts the peers according to their rating value.

5.1.1.2 Reading BGP Information

The Metering component calculates the generic rating parameters of IP addresses based on BGP information (see [D2.3] for details). It can read the BGP routing table from three different sources: 1) from a BGP router via SNMP, 2) from the SIS DB, or 3) from a local configuration file.

The reading of the BGP routing table and calculating the generic rating parameters include the following steps (see Figure 5):

1. The Metering queries the SIS DB for configuration parameters.
2. The SIS DB returns the configuration parameters.
3. According to the configuration parameters, the Metering reads BGP information either from a router, from the SIS DB or from a local configuration file. If the BGP

information shall be read from a router, the Metering sends SNMP queries to the Network (BGP router).

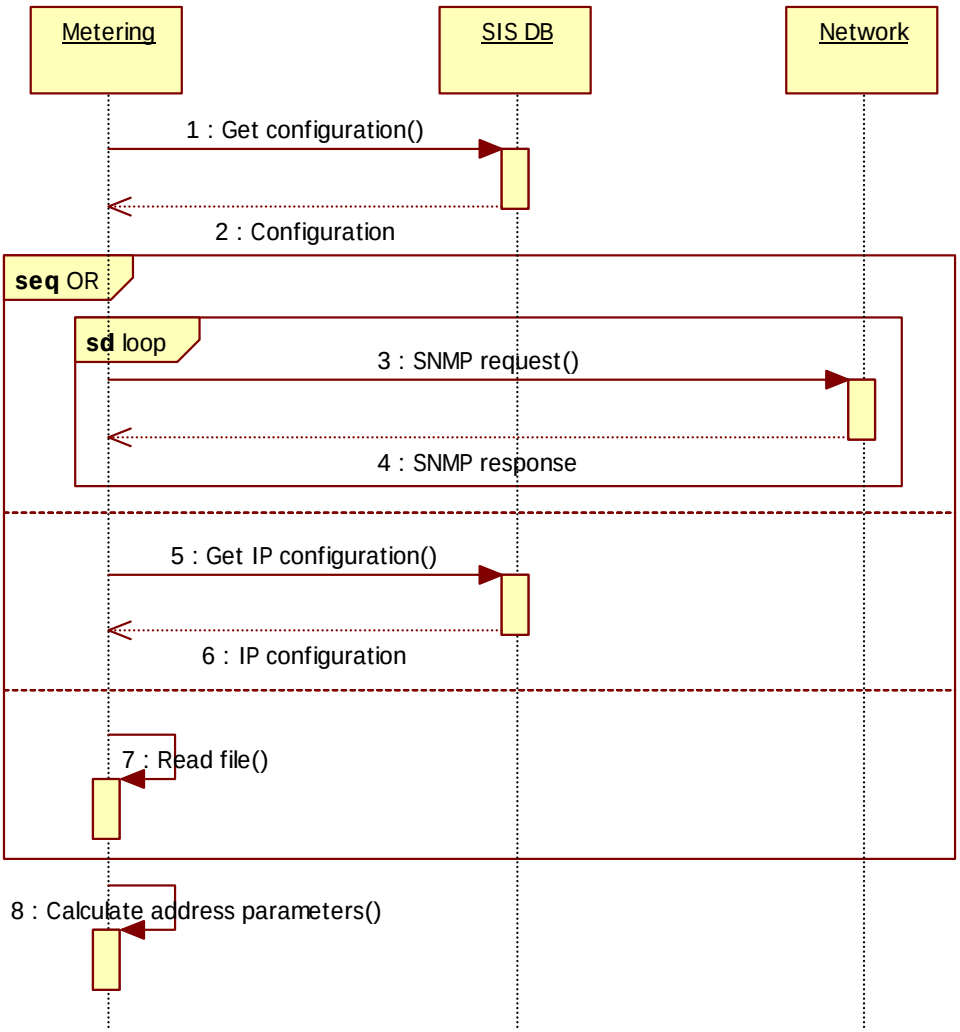

Figure 5: Reading BGP information

4. The Network (BGP router) sends back SNMP responses with the BGP information.
5. If the BGP information shall be read from the SIS DB, the Metering queries the SIS DB.
6. The SIS DB returns the BGP information.
7. If the BGP information shall be read from a file, the Metering read the local configuration file.
8. Based on the BGP information, the Metering calculates the generic rating parameters for each IP range [D2.3].

5.2 IOP

The SIS architecture components related to the IoP ETM are shown in Figure 6. Here the additional entity, the ISP-owned peer, is integrated into the architecture and exchanges swarm-related information with the SIS controller via the sis interface. On the other hand, the sis interface also allows retrieving information about locally active swarms from the overlay application.

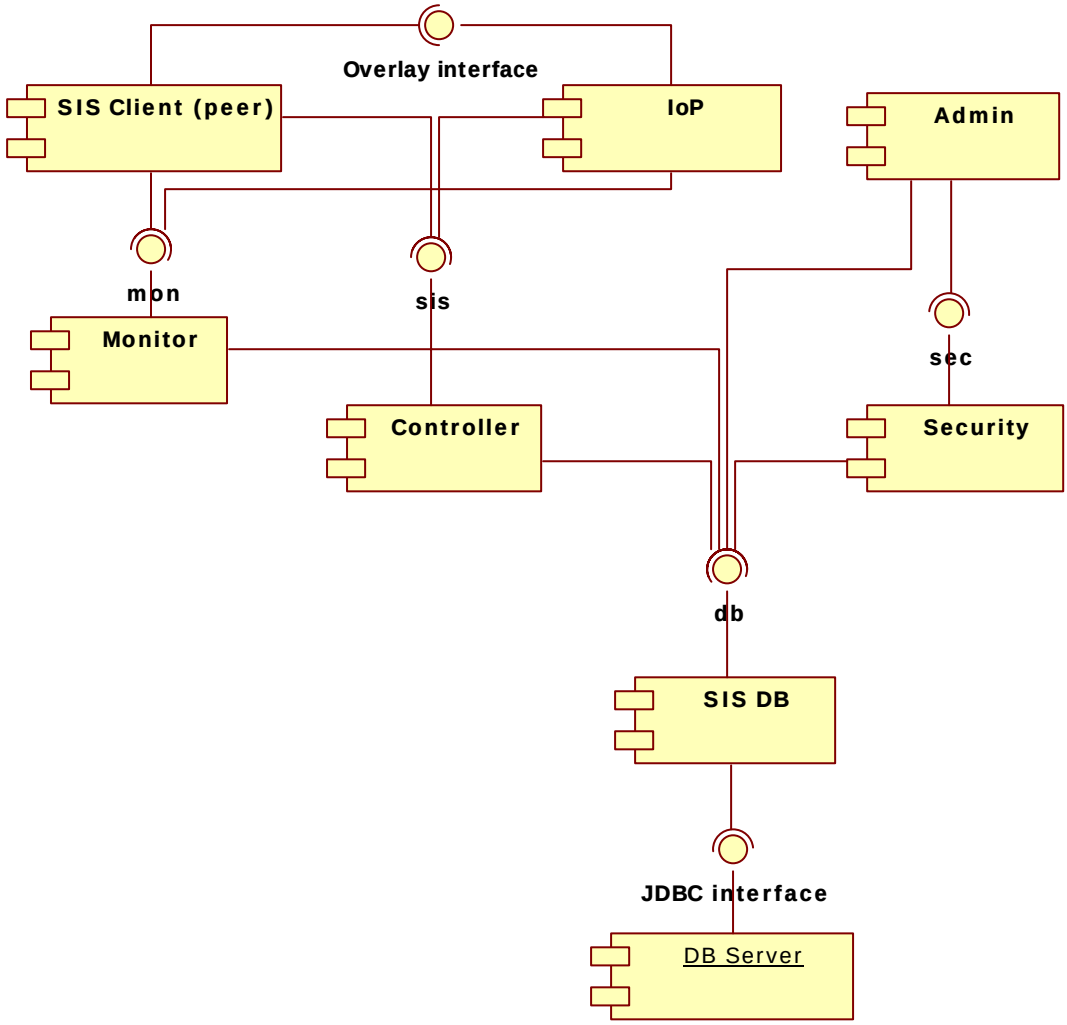

Figure 6: SIS architecture for IoP

5.2.1 Interactions

Relevant interactions cover the swarm statistics' reporting, the swarm selection, and the IoP configuration.

5.2.1.1 Reporting Swarm Statistics

The SIS Client sends swarm information to the Controller about swarms the peer participates in (see [D2.3] for details). The Controller stores this information and the IoP uses it to decide which swarms are popular. The swarm information is sent via SOAP.

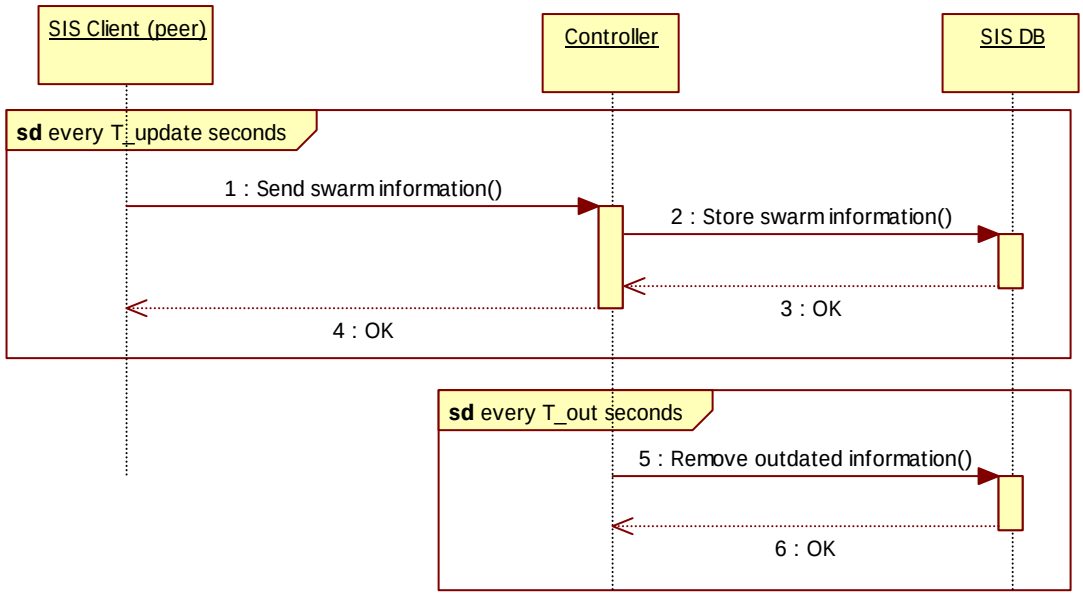

Figure 7: Reporting swarm statistics

The reporting of swarm statistics includes the following steps (see Figure 7):

1. The Client sends swarm information every T_update seconds to the Controller. The swarm information has to include:
    a. client IP and port
    b. for each swarm the peer participates in:
        i. torrent ID
        ii. torrent URL (containing the torrent file name)
        iii. file size
        iv. download progress in % (100% = seed)
2. The Controller checks whether the client IP is a local address (i.e. one from the ISP's domain). If yes, it stores the swarm information in the SIS DB by updating the swarm-info table and the client-info table. The Controller also updates the rate of each swarm, which are included in the report, in the swarm info table, where the rate is calculated by:
   rate = file_size * nof_local_leechers / (nof_local_seeders + 1)
   (nof_local_leechers and nof_local_seeders are taken from the client-info table by summing up the clients with progress < 100 and with progress = 100 for the given torrent ID)
3. OK if the swarm information is successfully stored in the SIS DB.
4. The Controller sends back an acknowledgement.
5. The Controller checks in every T_out seconds the entries in the SIS DB and removes outdated information. It removes clients that sent reports more than T_age seconds ago and removes swarms for which the last report was sent more than T_age seconds ago.

6. OK if the outdated swarm information is successfully removed from the SIS DB.

5.2.1.2 Swarm Selection

Based on the swarm information collected by the Controller, the IoP determines the swarms to join and to leave [D2.3]. The IoP retrieves swarm information via SOAP from the Controller and sends its swarm participation to the Controller.

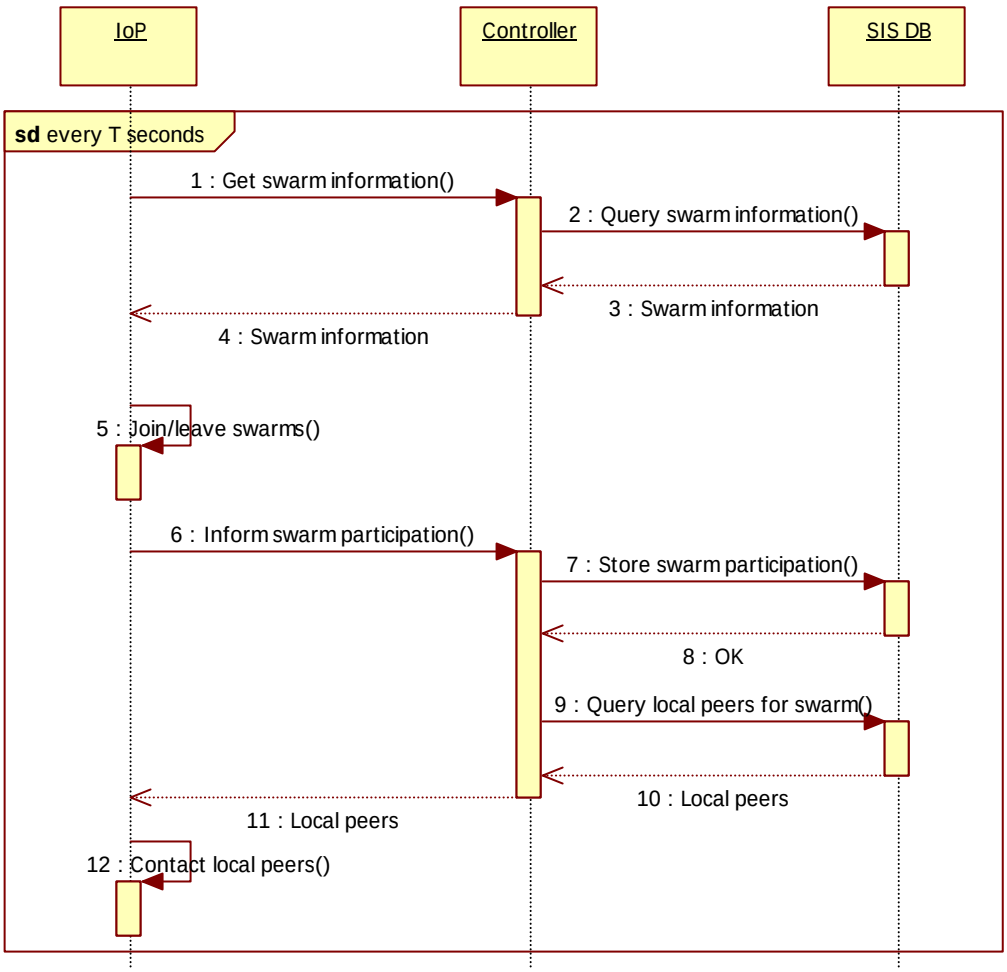

Figure 8: Swarm selection

The swarm selection includes the following steps (see Figure 8):

1. The IoP asks the Controller for swarm information about the top S swarms in which local peers are participating, where S = min{U/U_low, D/D_low} (U and D are the total upload and download bandwidth of the IoP, and U_low and D_low are the lower bounds for upload and download bandwidth provided for a single swarm). The message includes S, the number of swarms the IoP wants to get information about.
2. The Controller queries the SIS DB to get information about the top S swarms in which the IoP is not participating. The SIS DB stores this information in the swarm-info table. The Controller retrieves only the top S swarms, since the IoP participates in maximum S swarms.
3. The SIS DB returns the swarm information (entries of the swarm-info table).

4. The Controller sends the swarm information to the IoP, which includes a list of swarms (with up to S entries) containing the following information for each swarm:
    a. torrent ID
    b. rate (based on the formula provided before)
    c. torrent URL (The URL is required to get the filename of the torrent file from it. The file itself is located in a local directory on the IoP. Optionally, the URL could also be used by the IoP to download the torrent file if not available locally)
5. The IoP decides which swarms to join and leave. The IoP keeps participating in [x*S] top uploader swarms (or in all previous swarms if the number of previous swarms is less than [x*S]), where x and S are configuration parameters of the IoP (S = min{U/U_low, D/D_low}). The top uploader swarms are determined based on the total bytes that the IoP has uploaded to the peers of each swarm during the last time period T. The IoP fills the remaining swarm slots (up to S) with the highest rated swarms newly received from the Controller and joins these new swarms.
6. The IoP informs the Controller about its swarm decision and optionally asks for local peers in the selected swarms by sending the following information for each swarm it is participating in:
    a. torrent ID
    b. maximum number of peers (if 0, the IoP does not request local peers from the Controller)
    c. “include seeders” flag (whether to include seeders or not)
7. The Controller updates the swarm-info table (updates the IoP participation flag of all swarms, i.e. add/remove swarm participation) in the SIS DB according to the information received from the IoP.
8. OK if the swarm information is successfully updated in the SIS DB.
9. The Controller queries the SIS DB for local peer information if the *maximum number of peers* was set to greater than 0. The SIS DB stores the peer information in the client-info table.
10. The SIS DB returns the peer information (entries of the client-info table).
11. The Controller sends the peer information to the IoP. The reply includes per swarm the following information:
    a. torrent ID
    b. a list of IP address and port number pairs
12. The IoP contacts the local peers received from the Controller.

5.2.1.3 IoP Configuration

The configuration parameters of the IoP are stored in the SIS DB. The IoP can retrieve its configuration parameters via SOAP from the Controller. The IoP reads its configuration parameters at startup.

The following configuration parameters have to be supported:

- IoP operation mode (plain, combination, collaboration). Plain and combination imply static swarm selection and collaboration implies SIS-enabled swarm selection [D2.3].
- Enable/disable to connect to remote peers
- Number of unchoking slots per swarm
- List of local IP addresses
- For the SIS-enabled swarm selection:
    - T: time period in seconds to ask the SIS for new torrent statistics
    - U_low and D_low: Lower bounds for upload and download bandwidth in Mbit/s
    - U and D: Total upload and download bandwidth in Mbit/s
    - x: the percentage determining in how many previous swarms the IoP will keep participating (e.g., x = 0.9)
- Name/Address of the Monitor

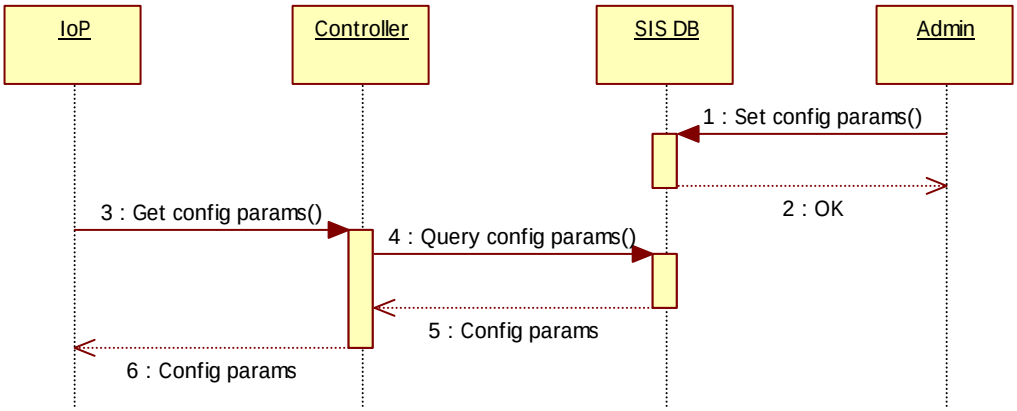

Figure 9: IoP configuration

The retrieval of the IoP configuration parameters includes the following steps (see Figure 9):

1. The operator sets the configuration parameters of the system using the Admin component, which stores the parameters in the SIS DB.
2. OK if the configuration parameters are successfully stored in the SIS DB.
3. The IoP requests its configuration parameters from the Controller.
4. The Controller retrieves the configuration parameters from the SIS DB.
5. The SIS DB returns the configuration parameters.
6. The Controller sends back the configuration parameters.

5.3 HAP

The SIS architecture components related to the HAP ETM are shown in Figure 10. This ETM mechanism makes use of the QoS Manager component to modify user access profiles, based on functionality provided by the underlying NMS. For this purpose, this component communicates with the network equipment of the ISP and the SIS Controller.

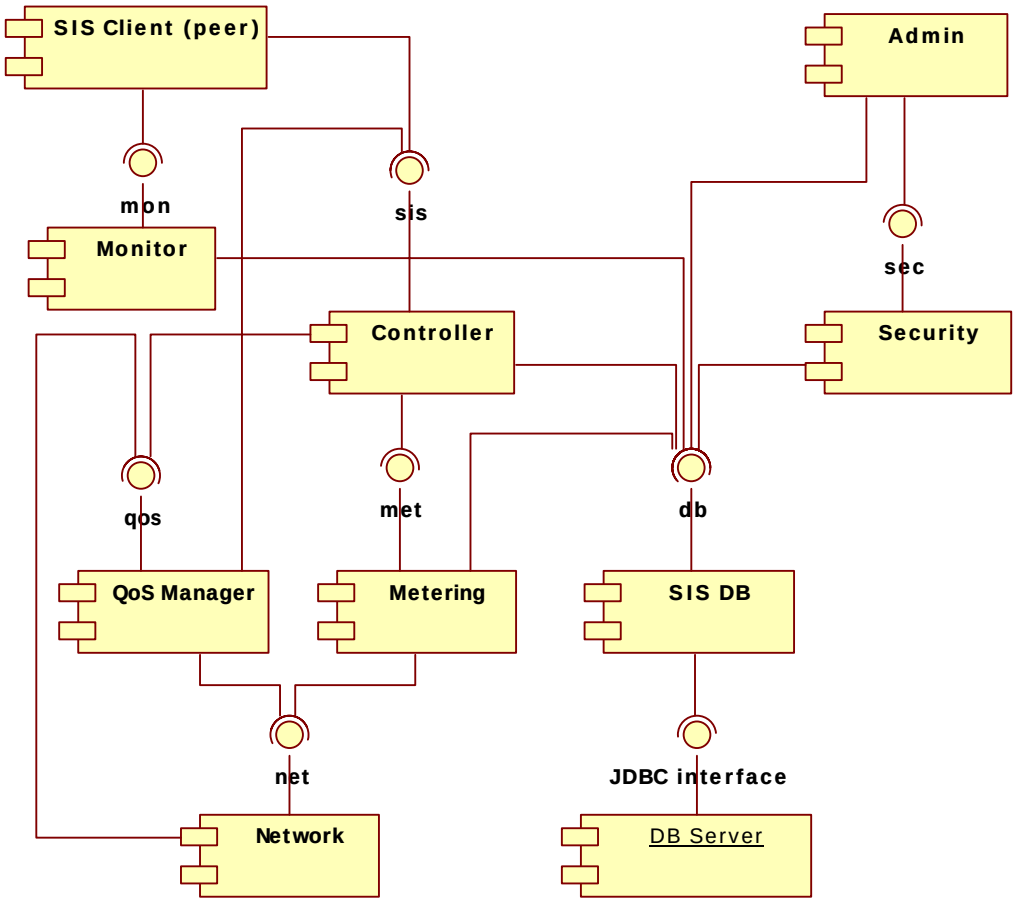

Figure 10: SIS architecture for HAP

5.3.1 Interactions

The key interactions cover the peer rating, the overlay reporting, the rating update, the HAP selection, and the bandwidth profile retrieval.

5.3.1.1 Peer Rating

The SIS Client can receive peer ratings from the Controller in the same way as in the case of the BGP-Loc ETM mechanism. The SIS Client – Controller interface is SOAP-based and it remains the same as in the case of BGP-Loc, but the Controller considers the HAPs in the rating calculation and assigns higher ratings to them if HAP is used together with BGP-Loc.

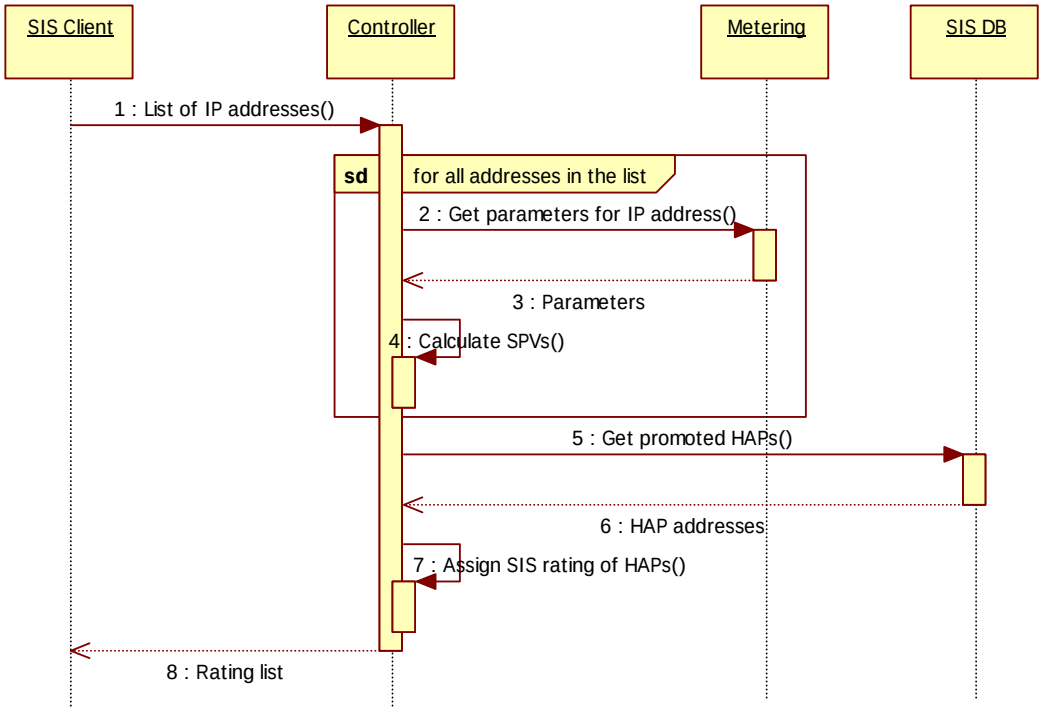

Figure 11: Peer rating

The retrieval of the SIS preference values includes the following steps (see Figure 11):

1. The Client sends the list of peers (IP addresses) to the Controller (same as in the case of BGP-Loc).
2. The Controller retrieves the rating parameters for each IP address in the list from the Metering.
3. The Metering returns the parameters.
4. Based on the parameters the Controller calculates the SIS preference values (SPV) for each IP address in the list.
5. The Controller queries the SIS DB for the list of current HAPs (promoted HAPs).
6. The SIS DB returns the list of current HAPs (IP addresses).
7. The Controller assigns the “HAP rate” to these peers in the list (if they are in the list). The HAP rate is higher than the rating for a normal local peer.
8. The Controller returns the rated list of peers (also including HAPs if any) to the Client. The Client contacts the peers according to their rating value.

5.3.1.2 Overlay Reporting

The SIS Client sends overlay information to the SIS Controller. The information includes the amount of download and upload traffic to other peers. The overlay information is sent over SOAP.

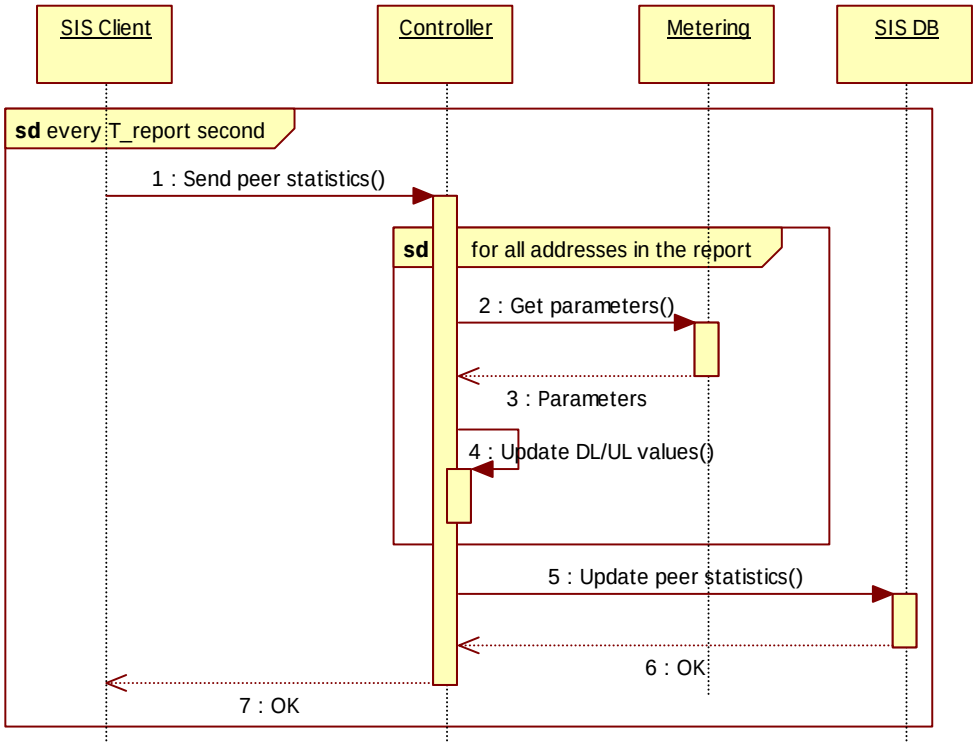

Figure 12: Overlay reporting

The overlay reporting includes the following steps (see Figure 12):

1. The SIS Client sends overlay reports to the Controller periodically at every T_report seconds (T_report is a configuration parameter of the client and it shall be in the range of a few hours). Additionally, in order not to lose information, the SIS Client also sends its final report, if the overlay application is terminated. The report always includes only the changes from the last report. The report includes the following information for each IP address, with which the client had communication in the last period and the amount of exchanged data is over a threshold (this threshold is a configuration parameter of the client):
    a. Upload traffic volume to the given IP address during time slot t (i.e. the last reporting period). (Unit: Megabytes [MB], Type: integer)
    b. Download traffic volume from the given IP address during time slot t. (Unit: Megabytes [MB], Type: integer)

    The report for all IP addresses has to be sent in a single SOAP message.
2. Based on the information received from the SIS Client, the Controller has to update the peer statistics and classify the traffic to local and remote traffic. Therefore, the Controller retrieves the rating parameters (including the local/remote flag) for each IP address in the report from the Metering.
3. The Metering returns the parameters.
4. The Controller calculates the total upload, total download, and local upload volumes of the peer during the last reporting period. (Unit: Megabytes [MB], Type: integer)

5. The Controller updates the peer statistics (total upload, total download, and local upload volumes) in the SIS DB. The new values are added to the values stored in the database. (Unit: Megabytes [MB], Type: integer)
6. OK if the peer statistics are successfully stored in the SIS DB.
7. OK if the report was successfully processed.

5.3.1.3 HAP Rating Update and HAP Selection

Based on the information received from SIS Clients, the Controller periodically calculates the HAP ratings of each peer and selects new HAPs. The Controller starts the calculation at a configured point in time (e.g., at 3:00 am in the morning).

The HAP rating update and HAP selection include the following steps (see Figure 13):

1. In every T_update seconds the Controller calculates the HAP rating for each known peer and determines new HAPs (T_update is in the range of 24 hours). Therefore, the Controller queries the SIS DB for the parameters (p1, p2, p3, AD, AU, D', U', RT) used in the HAP rating calculation [D2.3]:
    - p1, p2, p3: HAP rating calculation parameters (weights),
    - AD: the available download bandwidth an ISP is willing to assign to HAPs in total,
    - AU: the available upload bandwidth an ISP is willing to assign to HAPs in total,
    - D': the increase in download bandwidth given to each HAP,
    - U': the increase in upload bandwidth given to each HAP,
    - RT: the HAP rating threshold.
2. The SIS DB returns the parameters (p1, p2, p3, AD, AU, D', U', RT).
3. The Controller queries the SIS DB for the list of peers for which the database contains peer statistics. The list of peers is required to determine the upload bandwidth (B) of the Internet connection of each peer.
4. The SIS DB returns the list of peers (the IP addresses).
5. The Controller requests from the QoS Manager the bandwidth profile for all IP addresses retrieved from the SIS DB in the previous step. The Controller uses the qos.2 interface which is an internal API.
6. The QoS Manager gets the bandwidth profiles from the Network (see bandwidth profile retrieval).
7. The Network returns the upload bandwidth (B) for each IP address (Unit: Kilobits/second [Kbps], Type: integer).
8. The QoS Manager returns the bandwidth profiles to the Controller.
9. The Controller queries the SIS DB for the list of promoted HAPs. The list is required to determine the B(t) value in the HAP calculation. If the peer is not an HAP, then B(t) = B (the upload bandwidth of a normal peer). If the peer is an HAP, then B(t) = B+U'.

10. The SIS DB returns the list of promoted HAPs.
11. For every peer, for which the SIS DB contains peer statistics, the Controller performs the following: It queries the SIS DB for peer statistics received in the last update period and for the last HAP rating (R) of the peer (in a “get next” manner).

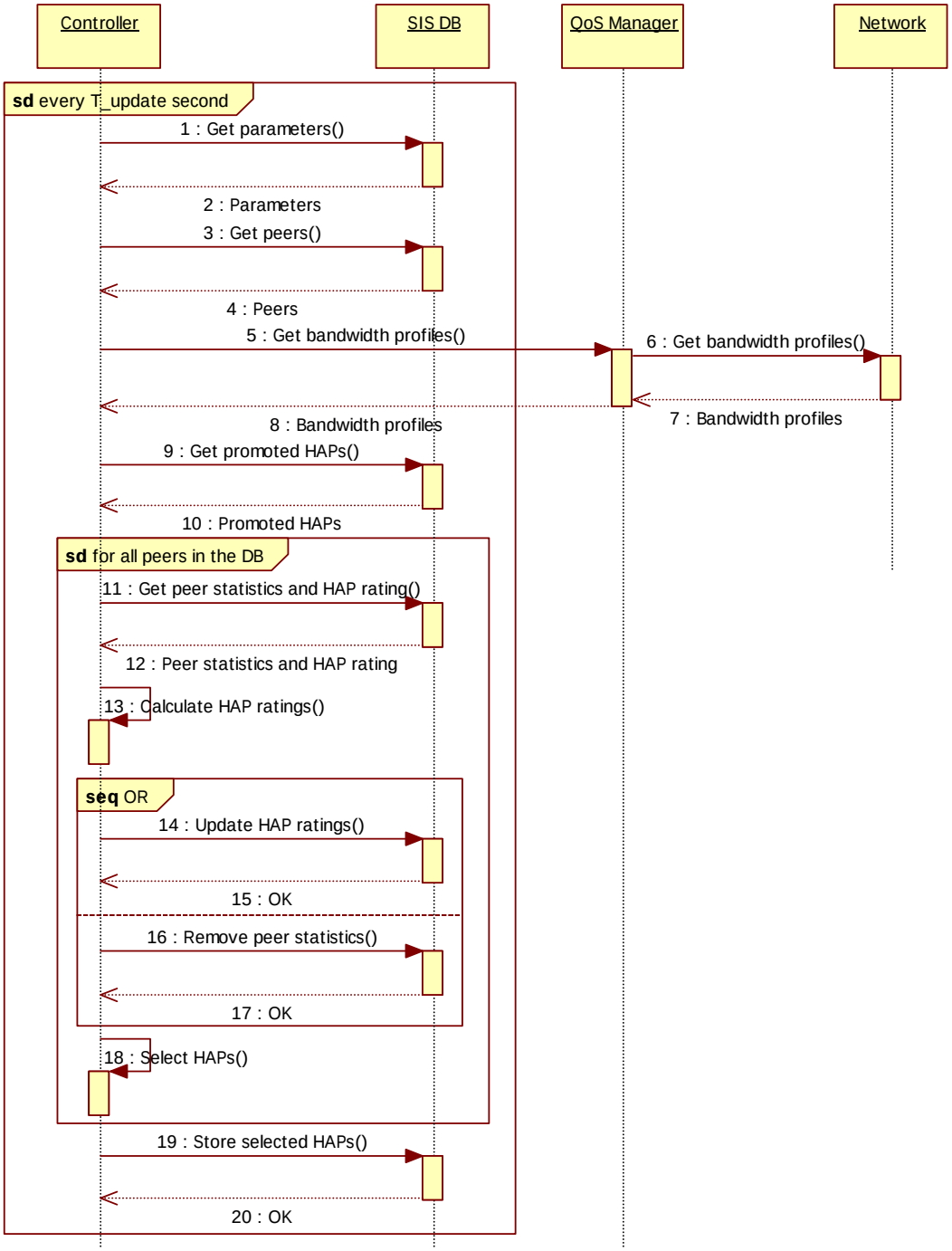

Figure 13: HAP rating update and HAP selection

12. The SIS DB returns the peer statistics (total upload U(t), total download D(t), and local upload L(t)) and resets their values in the database to 0. The SIS DB also returns R.
13. The Controller calculates the ratings R(t) for the last update period [D2.3]:

$$R(t) = p_1 \frac{1024 \cdot 8 \cdot L(t)}{d \cdot B(t)} + p_2 \cdot S(t) + p_3 \frac{1024 \cdot 8 \cdot U(t)}{d \cdot B(t)}$$, where

- L(t) is the local upload during the last update period (Unit: Megabytes [MB], Type: integer),
- U(t) is the total upload during the last update period (Unit: Megabytes [MB], Type: integer),
- S(t) is the seeding ratio in the last update period calculated by U(t)/D(t) (total upload divided by total download),
- B(t) is the upload bandwidth of the peer (Unit: Kilobits/second [Kbps], Type: integer),
- d is the duration of the update period (d=T_update) (Unit: seconds [s], Type: integer).

The Controller also calculates the total HAP rating R from R(t) and the previous value of R:

$$R_{new} = R(t) + e^{-1} R$$

14. If the total HAP rating is over the HAP rating threshold (RT), the Controller stores the new HAP rating of the peer in the SIS DB. The threshold is a configuration parameter. If RT is set to 0, all peer statistics will be stored in the SIS DB and none of them will be removed.
15. OK if R was successfully updated.
16. If the total HAP rating is below the HAP rating threshold (RT), the Controller removes the peer from the peer statistics.
17. OK if the peer was removed.
18. The Controller puts the peer in a temporary list if it belongs to the top N peers with the highest rating R. Thus, after updating the HAP rating of all known peers, the Controller will have the top N peers. N is a configuration parameter. If N is set in the SIS DB, the configured value is used. If N is not set in the SIS DB, N is calculated by

$$N = Min(\frac{AD}{D'}, \frac{AU}{U'})$$, where

- AD is the available download bandwidth an ISP is willing to assign to HAPs in total,
- AU is the available upload bandwidth an ISP is willing to assign to HAPs in total,
- D’ is the increase in download bandwidth given to each HAP,

- U' is the increase in upload bandwidth given to each HAP. (They are all configuration parameters)

19. The Controller stores the selected HAPs in the SIS DB as new HAPs (removing the previous list of new HAPs).
20. OK if the HAPs were successfully stored.

5.3.1.4 Bandwidth Profile Retrieval

The QoS Manager can retrieve the bandwidth profiles (the upload bandwidth) assigned to IP addresses/customers via SOAP from the Network. The bandwidth profiles are retrieved during the HAP rating calculation (see above). The Network (e.g., the billing system of PrimeTel) has to provide this information.

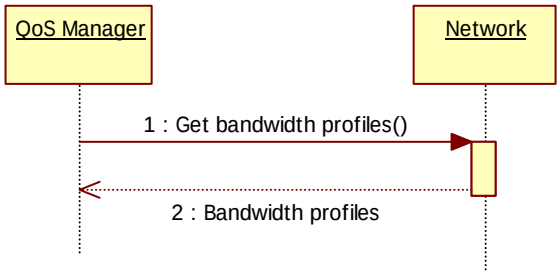

Figure 14: Bandwidth profile retrieval

The retrieval of bandwidth profiles includes the following steps (see Figure 14):

1. The QoS Manager sends a request to the Network (e.g., the billing system of PrimeTel) to retrieve bandwidth profiles. The request contains a list of IP addresses (Type: string).
2. The Network sends back the bandwidth profile to each address in the request. The bandwidth profile contains the upload bandwidth (B) assigned to the IP address/customer (Unit: Kilobits/second [Kbps], Type: integer).

5.3.1.5 HAP Retrieval

The Network (e.g., the Network Management System of the ISP) retrieves the list of IP addresses that shall be promoted to HAP and the list of IP addresses that shall be demoted from HAP. The Network can retrieve this information from the QoS Manager via SOAP. The Network changes the traffic profiles (DL and UL bandwidth) of these peers accordingly.

The HAP retrieval includes the following steps (see Figure 15):

1. In every T_profile seconds the NMS of the ISP asks for the list of HAPs to be promoted and demoted (T_profile is in the range of 24 hours).
2. The QoS Manager forwards the query to the Controller.
3. The Controller queries the SIS DB for the list of new and promoted HAPs. New HAPs refer to the HAPs calculated in the last update as top N peers. Promoted HAPs refer to HAPs that have currently a higher UL/DL configured.

4. The SIS DB returns the list of new and promoted HAPs.
5. The Controller determines the list of HAPs to be promoted (P) and to be demoted (D):

   P = the set of new HAPs – the set of promoted HAPs

   D = the set of promoted HAPs – the set of new HAPs
6. The Controller stores the new HAPs as promoted HAPs in the SIS DB (removing the previous list of promoted HAPs and leaving the list of new HAPs unchanged).
7. OK if the promoted HAPs were successfully stored.
8. The Controller returns the list of HAPs to be promoted (P) and demoted (D).
9. The QoS Manager returns the list of HAPs to be promoted (P) and demoted (D).

The Network updates the traffic profiles of the specified peers, i.e. for promoted HAPs uplink = uplink + U' and downlink = downlink + D'.

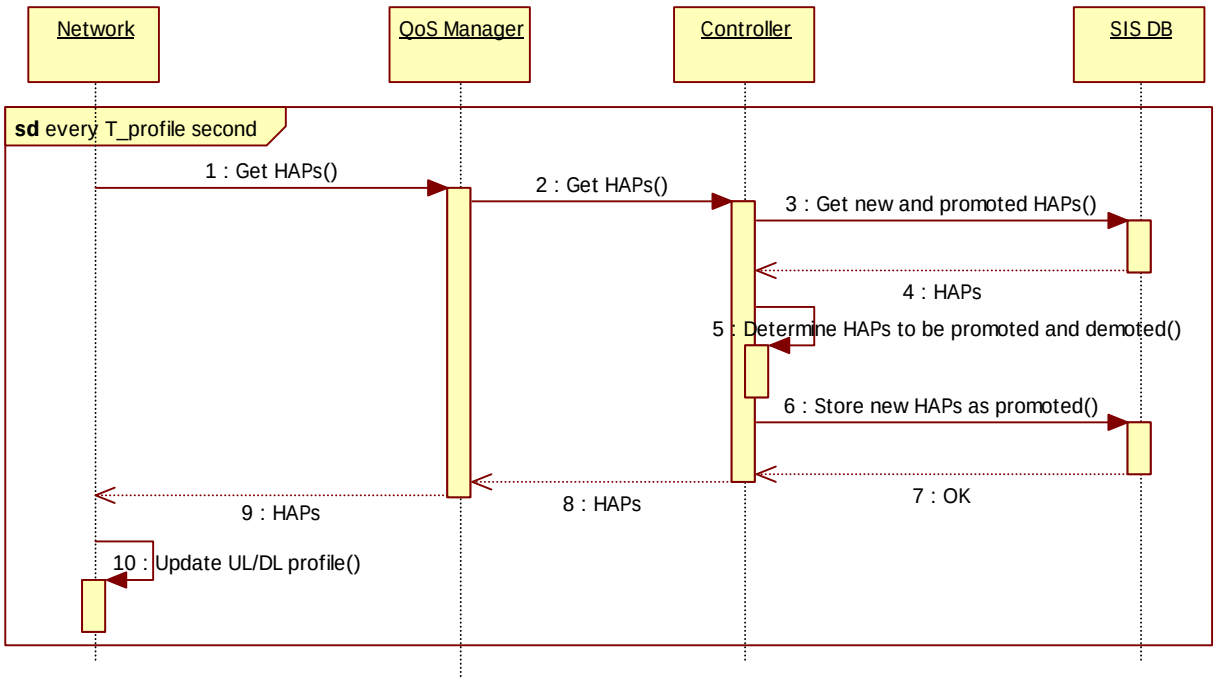

Figure 15: HAP retrieval

6 Requirements

The SmoothIT project has defined both functional and non-functional requirements. They were defined in Deliverable D1.1 [D1.1] and further refined during the process of designing and detailing the system architecture.

6.1 Functional Requirements

Table 6-1 shows the main functional requirements identified for the SmoothIT architecture.

Table 6-1: Functional Requirements

ID	Requirement	Core req.	Add. req.
R.1.	Improving P2P application performance while reducing the network traffic: SmoothIT will create a win-win-win situation for the involved players: end-users, ISPs and, possibly, overlay providers. As an example, end-users may benefit through selecting local peers to connect to, while ISPs can benefit by network status gathering and subsequent traffic optimization and shaping. SmoothIT will come up with overlay operation strategies that improve on the current practice by employing these two techniques simultaneously.	X	
R.2.	Incentive-compatibility: The solutions provided by SmoothIT will be incentive compatible in the sense that it will be in the best interest of all involved players to behave according to the rules of the proposed SmoothIT protocols.	X	
R.3.	Support of different overlay applications: The SIS shall provide an open service that is accessible by different P2P applications.	X	
R.4.	Interface supporting various optimization schemes: The interface between the SIS and the overlay application shall provide means to specify the application scenario and the respective parameters. Due to the various incentives of ISPs, overlay providers, and end-users, the SIS shall provide several services (*e.g.*, “Throughput Optimization”, “QoS enhancement”) that could be classified into *free* and *premium* (charged) network services.	X	
R.5.	QoS support: The SIS shall support QoS for network services and it shall be able to configure network resources.	X	
R.6.	Different modes of operation: The SIS shall be able to operate in two different modes: user anonymity mode for free services and user aware mode for premium services.		X

R.7.	Inter-domain support: The SIS deployed in different ISPs shall be able to interact with each other. SIS elements in different ASs may communicate with each other in order to get the overall view of a communication in respect of the optimization parameters specified.		X
R.8.	OAM (Operation and Management) support: The SIS shall be able to interact with the OAM processes of the ISP.		X
R.9.	Mobile network support: The above requirements should also be valid in the context of a cellular network operator, which is characterized by the following key properties: node mobility, heterogeneity of nodes and link capacities, and presence of shared medium.		X

6.2 Non-functional Requirements

Table 6-2 shows the non-functional requirements identified for the SmoothIT architecture.

Table 6-2: Non-Functional Requirements

ID	Requirement	Core req.	Add. req.
R.10.	Easy deployment: It shall be easy to extend existing overlay applications with the functionality of the SIS and it shall be easy for ISPs to deploy the SIS in their networks.	X	
R.11.	Extensibility: The SIS shall be extendible to support new overlay applications, new optimization attributes, and new metrics (both application-driven and provider-driven).	X	
R.12.	Scalability: The SIS shall be scalable to support a large end-user population.	X	
R.13.	Efficiency: The operation of SIS shall be efficient in terms of communication (bandwidth) overhead, storage consumption, and processing requirement.	X	
R.14.	Robustness: The SIS shall be robust against malicious behavior and against dynamic behavior (churn of peers). It shall be also fault tolerant.	X	
R.15.	Security: Secure communication between SIS entities and between SIS and overlay application shall be supported, providing message origin authentication, data integrity, and data confidentiality. Any data storage in the system shall provide data integrity, confidentiality, and authentication.	X	
R.16.	Standard compliance: The SIS shall use and based on standard protocols where applicable.	X	

R.17.	Transparency: The SIS shall not apply Deep Packet Inspection (DPI).	X	
R.18.	Data privacy and legislation/regulation: The SmoothIT architecture needs to provide interfaces for regulation aspects, such as data retention, and it has to address data privacy concerns, which are determined by the European Directives on Security.		X

6.3 Addressing the Identified Requirements

R1: Simulations and the internal trial show performance improvements on both sides. They at least demonstrate a “win - no loose” situation, i.e., the network operator benefits is better off, while the user is unaffected.

R2: Incentive-compatibility has been demonstrated by a number of simulations. These simulations show that it is always in the best interest of the peers, i.e., users to follow the recommendations of the employed ETM mechanism.

R3: There is nothing in the architecture specific to only one application, i.e., not general enough to be applicable to any overlay application. Proposed mechanisms reside on algorithms and protocols which are application dependent, but that is not the case with the architecture as a whole.

R4: The interface between overlay applications and SIS was intentionally left general enough to accommodate transfer of various parameters back and forth.

R5: The architecture is explicitly defined with the QoS support in mind. QoS components and functions are included in release 3.0 of the SmoothIT software.

R6: We did not develop mechanisms that rest exactly on these requirements. We haven’t distinguished the service to free and premium. We assume that the implemented services can apply to all peers. Anonymity is considered throughout all mechanisms, with the exception of the HAP ETM mechanism where the system needs to know the customer, i.e. its IP address, in order to reward him. However, if required, the developed architecture can support those types of services easily.

R7: This requirement is explicitly addressed by the “Inter-SIS” effort submitted to IETF, ALTO: [Inter-SIS-1, Inter-SIS-2]. Although the current architecture can support the inter-SIS mechanism, the specific mechanism was not implemented.

R8: The HAP mechanism interacts with the OAM by retrieving customer profiles and suggesting customers to have their links upgraded, for a short-to-medium time scale. However, the actual implementation of those services rely on the ISP's side. No other of the implemented mechanisms required interaction with the OAM system.

R9: This requirement is explicitly addressed by the efforts documented in [D2.4]. The results documented here show important differences between applying locality in mobile and fixed networks.

R10: According to our experience from the internal trial, the deployment is quite easy. Configuration is also straightforward.

R11: As we are not bound to specific overlay applications, extensibility to fit the needs of other applications is easily achievable. Moreover, the current architecture can be extended to support any new ETM mechanism, due to its modular design and the well-defined functionality of each component.

R12, R13 and R14: The exact information on how well we meet these requirements will be provided after the external trial in September 2010. The operation of the system will be monitored and the exact statistics showing how the system fulfills these requirements will be collected. They will be documented in a deliverable after the trial.

R15: To meet this requirement, the security component was designed. SIS server supports the most important security services: authentication, confidentiality, data integrity, access control, non-repudiation and others. Security services were tested in the admin web interface but they are available in all JBoss external interfaces such as JMX console or web services.

R16: SIS uses standard protocols. In fact, it uses SOAP for the request/response transactions and this makes the integration from any program easy.

R17: SIS behavior is not based on DPIs. The clients that want to use the SIS explicitly ask the SIS for ratings of other peers. Also, the mechanisms use peer-provided anonymous overlay statistics in order to estimate the status of the underlay.

R18: During the design process of SIS architecture we have taken into account documents such as: Directive 95/46/EC, Directive 97/66/EC, Directive 2002/58/EC and Directive 2006/24/EC (especially during considerations about the content analyzer—optional and not implemented component).

7 Design Space and Implemented ETM Mechanisms

In this section we comment on the initial ideas on the design spaces from which the ETM mechanisms have emerged. These design spaces were described in [D3.1], prior to the specification of any concrete ETM mechanism and detailed description of their architecture. Hence, it is important to revisit the initial ideas and comment on how the implemented ETM mechanisms map to them.

7.1 Design Space Options

Here, we briefly remind the originally presented design space choices, describing their main characteristics and specified functionality.

Honey Pot: this design option mainly describes the functionality required for an over-provisioned peer to serve the ETM objectives. The system gathers P2P statistics from overlay entities, like the tracker, and filters the information related to the specific domain. Thus, the system can estimate the popularity of certain torrents and instruct the super-peer to join the respective swarms. The super-peer has augmented bandwidth and storage capabilities and its participation to a swarm will help for a faster dissemination of the content inside the domain, while reducing the incoming traffic volumes traversing the inter-domain link(s). Hence, the Honey Pot resembles a smart cache. A Content Analyzer module was also foreseen in order to filter copyrighted content. Finally, the system interacts with the underlying NMS so that it considers the network conditions in its decision on where to deploy the super-peer instances.

Control Freak: in this case the system monitors the overlay traffic and provides incentives to the end users in order to comply with the ETM objectives. In some cases, the system adjusts the connection speeds of the peers by means of traffic shaping. The objective is to reward or punish peers in order to reach an economically sound state, in terms of interconnection expenses (for the ISP) and overlay performance (for the end-users). The peers (end users) send the “price/performance” ratio preference to the system, so that it can be considered during the decision taking phase. Combing this information with congestion indicators and network status, the system can propose to the domain’s local peers which remote peers to connect to in order to achieve a dual goal: offer the maximum overlay performance for the peers while keeping low the inter-domain costs for the ISP. Traffic shaping mechanisms and flexible pricing schemes are the means to bias the user behavior in order to reach the desired objectives.

Block Party: this approach describes a way of two different ISPs to exchange information in order to fine-tune and potentially optimize the local decisions with respect to the ETM objectives of the deployed solutions. The information exchanged depends on the deployed systems. In case of two neighbor ISPs having deployed the Honey Pot approach, such information could include the content that was decided to be cached by the two systems, in order to be able to share cache contents and optimize the offered services.

Optimal Anarchy: this design option decentralizes the functionality described in the Control Freak case. Such a system achieves the rewarding and punishment of peers in a distributed manner. There are no longer any central entities or dedicated devices orchestrating the entire function of the system. The ETM mechanism is distributed in the end-user equipment (P2P applications) and the core network elements (routers). A hierarchy of ETM decisions is now formed, following the topology of the Internet structure.

These decisions consider again the “price/performance” ratio preferences of the users, the congestion levels of the inter- and intra-domain links and the inter-domain charging levels.

7.2 Mapping of ETM Mechanisms to Design Space Options

In this section, an analysis of how the implemented ETM mechanisms, namely the BGP-Loc, the IoP and the HAP, map (fully or partly) to the original design spaces is presented. The original design space provided some options that helped forming the final ETM mechanism. However, not all ETM mechanism were inspired by a single design space option, i.e., a one-to-one mapping is not always the case. To make the mapping clearer,

we first identify the common functionality offered by the specified ETM mechanisms and the design space options. We then identify the overlaps in the architectural design and we finally focus on the differences (if any) of the interfaces between the various components.

For some ETM mechanisms the mapping is quite straightforward, for others deeper understanding of the ETM mechanisms and the design spaces is required. To gain more insight on the ETM mechanisms, please consult [D2.3] for the specification and Section 5 of this deliverable for the design and [D3.3] for the implementation details.

7.2.1 BGP-Loc

The concept of the BGP-Loc ETM mechanism is very close to the idea of the Control Freak. In fact, similarly to a subset of the functionality described in the Control Freak specification, the core function of BGP-Loc ETM mechanism is to rate lists of neighbor peers according to certain BGP metrics. The idea of peer rating is actually an enrichment of existing overlay metrics (e.g., throughput, etc) with underlay statistics (e.g., hop count, preference, etc). However, BGP-Loc is not an instance of the Control Freak, in the strict sense, since it does not enforce any decisions to the client. It just proposes some “good” neighbors from the ISP’s point of view, which could improve the overlay performance and at the same time decrease the inter-domain charges. Due to it’s non-intrusive nature, the BGP-Loc mechanism does not use the rest of the functionality proposed by the Control Freak approach.

In Figure 16 we compare the two architectural designs and the respective components and interfaces involved. It is obvious that the functionality offered by the BGP-Loc mechanism is only a subset of that expected in the Control Freak option. The main reason for this is that we wanted the first ETM mechanism implemented to be close to the state of the art in the area of overlay traffic localization. A tighter interaction with the underlying NMS is achieved in later ETM mechanisms, i.e., the HAP mechanism.

7.2.2 IoP

The IoP ETM mechanism is perhaps the most straightforward case of mapping to a design space option. Comparing the specifications of the Honey Pot and the IoP, it is made clear that the former fully describes the implemented functionality of the latter. The ISP-owned peers act as a non-transparent caches acting as super-peers, with increased access speeds and dedicated storage. Their purpose is to download the content inside the domain as quickly as possible and then distribute it to the local peers. Modified overlay procedures like neighbor selection and unchoking, as well as certain configuration parameters ensure that the ETM goals are achieved.

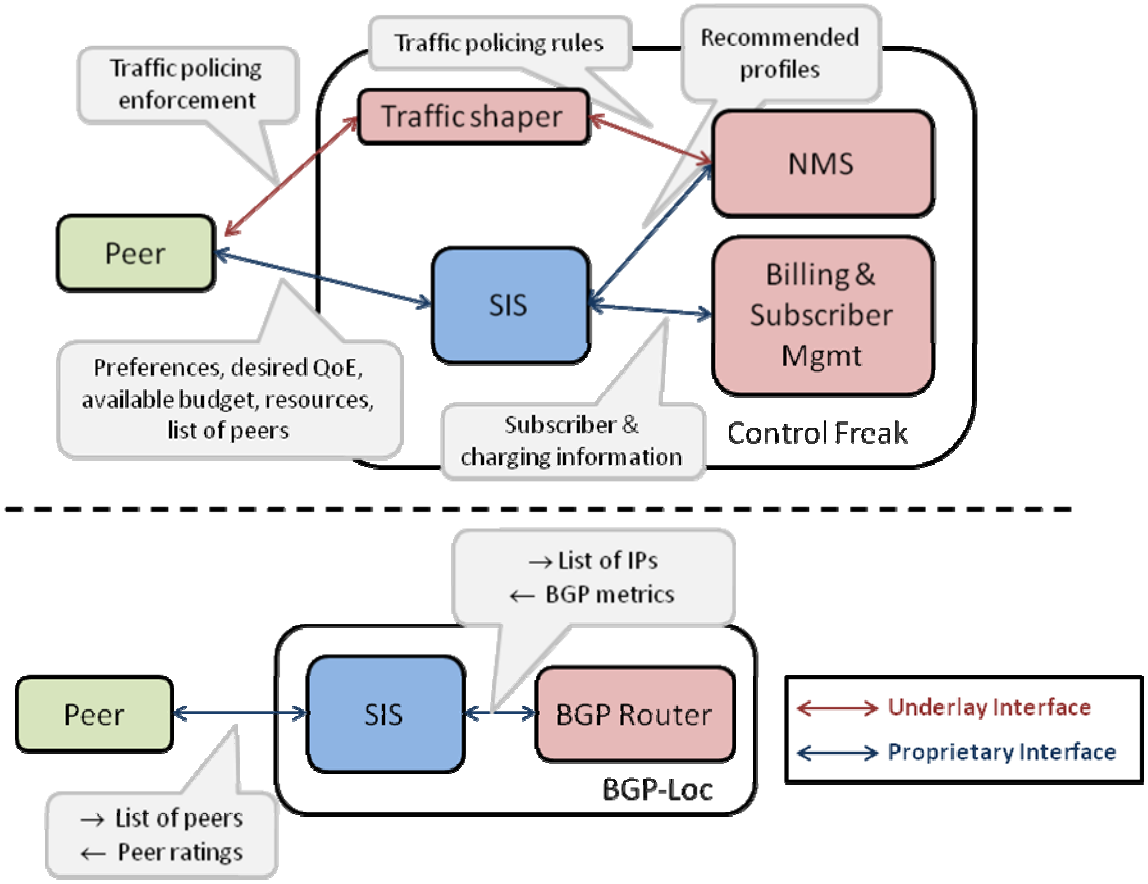

Figure 16: Mapping of BGP-Loc ETM mechanism to the Control Freak design option

However, there are some differences with the originally described concept of the Honey Pot. First of all, the IoP does not interact with the underlying NMS to get a view of the network conditions. In the specification of the IoP mechanism, it is assumed that the placement of the IoP(s) is decided by the ISP and is not dynamically adapted to the network conditions. Another point of difference is that the IoP does not include a Content Analyzer component. It is assumed that the ISP that deployed the IoP has a business relationship with certain content providers/distributors, and hence the IoP participates in the swarms of the specific, legal, content by acquiring a priori (through offline communication) the required torrent files. Note that even if the IoP has the torrent files of all the content available for P2P distribution, it does not join every swarm but rather waits for the SIS to recommend the most popular torrents. In close connection with the previous point, is the fact that the IoP does not interact with any tracker to estimate the popularity of the content. The main reason is again the legality issues that arise with copyrighted content. On the contrary, the SIS receives from the local peers a report about their overlay activity and uses such information to estimate the content popularity and instruct the IoP which content to cache.

In Figure 17 we present the architectural diagrams of both the Honey Pot design option and the IoP ETM mechanism. All the required components are included, along with the interfaces between them. It becomes obvious that there is a high overlap in the architectural diagrams of the Honey Pot and the IoP. Due to the reasons mentioned earlier, some of the Honey Pot components were not used, like the NMS and the tracker,

while some new interfaces were introduced in the case of the IoP, i.e., the overlay activity reporting.

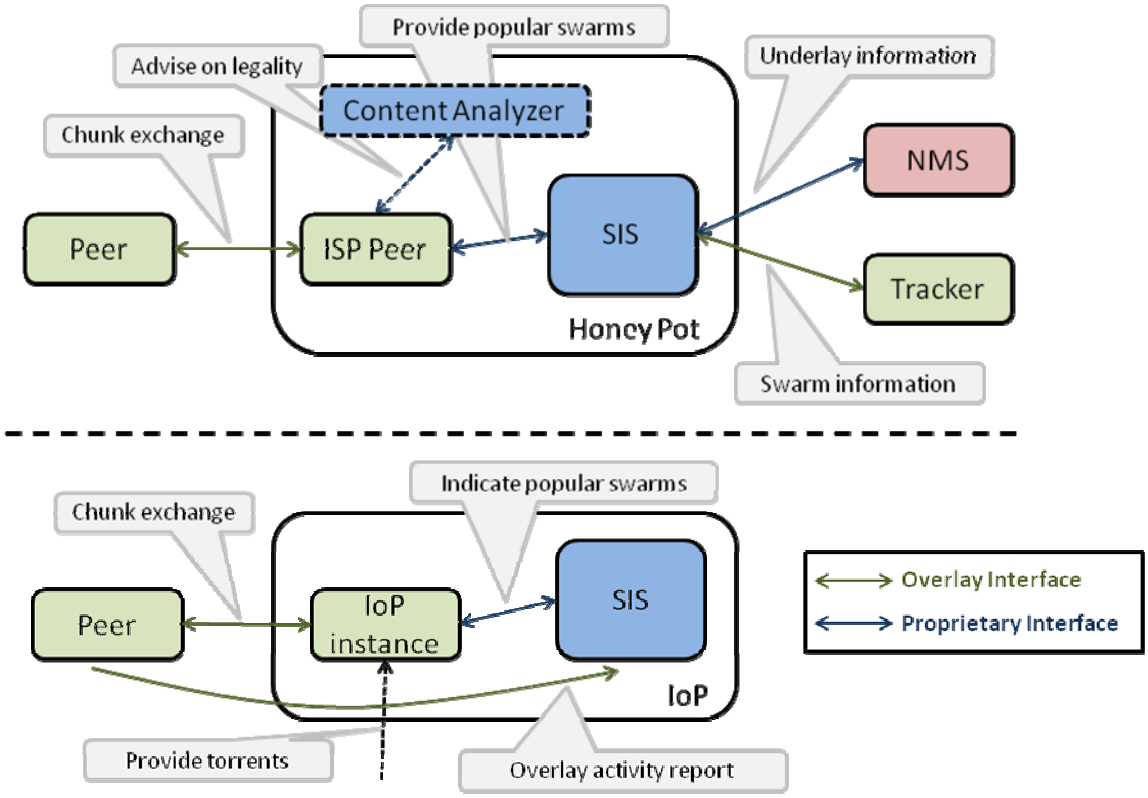

Figure 17: Mapping of IoP ETM mechanism to the Honey Pot design option

7.2.3 HAP

Conceptually, the HAP ETM mechanism can be considered as a decentralized approach of deploying super-peers inside the ISP's domain. Hence, one could claim that the HAP and IoP mechanisms are closely related. However, in terms of architectural design and functionality offered, the HAP mechanism is close to the Control Freak option, with slightly different objectives. The goal of the HAP mechanism is to reward peers that are active in swarm and more precisely those that serve other local peers, i.e., peers from the same domain. To do so, the ISP promotes those peers to HAPs by increasing their access speed (by means of traffic shaping or other NMS techniques, like NGN dynamic profile updates functions). The HAPs can now download content faster but also offer it faster to other local peers. The activity of local peers (HAPs or standard ones) is monitored so that standard peers can be promoted to HAPs and HAPs demoted to standard peers if their behavior after the bandwidth boost is not the one that was expected. This update is done once a day, or even more often, if the NMS procedures allow doing so.

The main difference with the Control Freak is that the HAP mechanism does not consider any QoS or other requirements from the peer, nor it accepts any lists of potential overlay neighbors for rating (unless combined with the BGP-Loc mechanism, as described in the following subsection). Moreover, the contract between the customer and the ISP, as well as the respective charges, is not affected. In fact, the allocation of more resources to the end users is done free of charge, as an incentive for the peers to act in accordance with

the ISP's objectives. That is to remain online for a long time so as to serve other local peers. In Figure 18 we show the architectural diagram of the HAP. When compared with the upper diagram of Figure 18, one can observe the resemblance between the two architectural approaches.

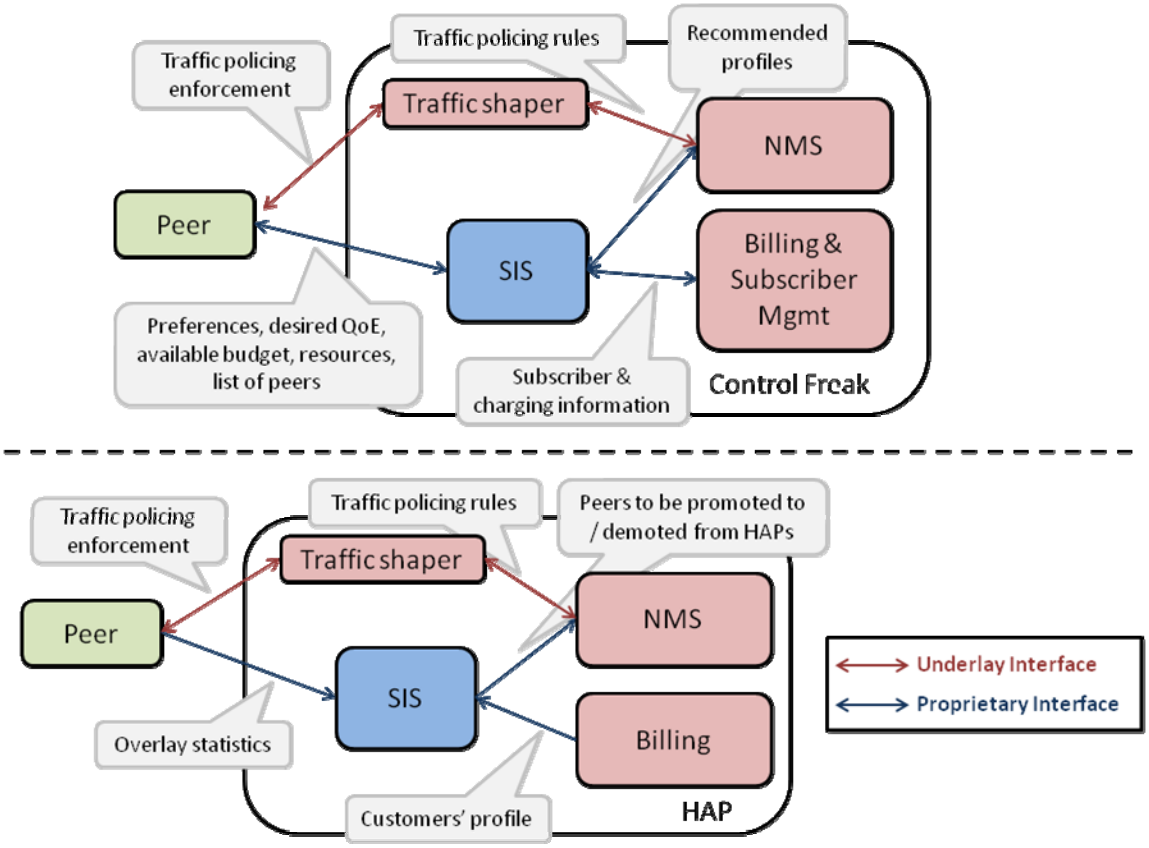

Figure 18: Mapping of HAP ETM mechanism to the Control Freak design option

As already mentioned, the HAP mechanism requires a close collaboration with the underlying NMS system. The overlap with the original Control Freak diagram is obvious. Note however that the objective of the HAP mechanism is not to directly fulfill the end users' QoE/QoS requirements but to indirectly improve the "health" of swarm, keeping the traffic as local as possible.

7.2.4 Combinations of ETM Mechanisms

In [D2.3], as well as in the external trials description to be included in the forthcoming WP4 deliverables, we have identified the possibility of combining certain ETM mechanisms so as to achieve a clearer TripleWin situation. Since all three implemented ETM mechanisms are disjoint, they can be easily combined without any further implementation efforts required. For example, BGP-Loc can be combined with either the IoP or the HAP mechanisms so as to introduce further gains to the end user. When considering the combination of BGP-Loc and HAP, one can immediately observe the almost full mapping of the implemented architecture with the initially proposed design space option of the Control Freak.

7.3 Discussion

We have seen in this section, how the implemented ETM mechanisms map to the originally described design space options. In fact, the initially widely-described design spaces encompass much of the functionality that the final ETM mechanisms have realized. As it was expected, the ETM mechanisms implement a subset of the available functionality. The mapping between ETM mechanisms and design space options was presented in terms of used components and interfaces. Additionally, possible combinations of ETM mechanisms also fall under the design space options. This fact indicates that the design phase, from its early stages till the final outcome, was concrete and extendable enough to include any originally unspecified characteristics of the desired architecture.

8 Summary and Conclusions

This deliverable presents the final SmoothIT system architecture. It gives a high level view on the components present in the system, their functionalities and interfaces among them. The document is structured not only to explain how the solutions developed throughout the project work but also to serve as a guide for development of future mechanisms, pointing where to place specific functionalities, how to configure them and also providing skeleton implementation of the components needed by those mechanisms. With respect to this, our explanation of how our selected ETM mechanisms map to the architecture is particularly valuable.

The deliverable also presents a look back at the project and revisits and reevaluates choices we made at earlier stages. No serious change has been made with the initial architecture, which demonstrates that we understood the problem we were dealing with as early as in the starting phase of the project. We analyzed how well we fulfilled the requirements defined earlier in WP1. The main outcome of the analysis is that most of the requirements were fully met. However, there are a number of them that still need to be tested during the external trial. Preparations for those tests are underway. The results of the tests will be reported in a later deliverable.

We also outlined how the implemented ETM mechanisms map to the originally described design space options. As it was expected, the ETM mechanisms implement a subset of the functionality specified in the design space.

9 References

[p2p-next] P2P Next EU/ICT project, http://www.p2p-next.org/

[D1.1] SmoothIT Project: *Requirements and Application Classes and Traffic Characteristics (InitialVersion)*; Deliverable D1.1, July 2008.

[D2.2] SmoothIT Project : *ETM Model and Components (Initial Version),* Deliverable D2.2, December 2008.

[D2.3] SmoothIT Project : *ETM Model and Components and Theoretical Foundations (Final Version),* Deliverable D2.3, October 2008.

[D3.1] SmoothIT Project : *Economic Traffic Management Systems Architecture Design (Initial Version)*, Deliverable D3.1, October 2008.

[Inter-SIS-1] Z. Dulinski, M. Kantor, W. Krzysztofek, R. Stankiewicz, and P. Cholda, “Optimal Choice of Peers based on BGP Information,” Proceedings of 2010 IEEE International Conference on Communications (ICC), May 2010.

[Inter-SIS-1] Z. Dulinski, R. Stankiewicz, P. Cholda, P. Wydrych, B. Stiller, “Inter-ALTO communication protocol”, submitted as IETF ALTO draft, draft-dulinski-alto-inter-alto-protocol-00.txt, June 2010.

[D2.4] SmoothIT Project: Performance, Reliability, and Scalability Investigations of ETM Mechanisms, Deliverable D2.4, August 2010.

Acknowledgements

This deliverable was made possible due to the large and open help of the WP3 team of the SmoothIT team within this STREP, which includes besides the deliverable authors the internal reviewers and Burkhard Stiller as well. Many thanks to all of them!