Simple Economic Management Approaches of Overlay Traffic in Heterogeneous Internet Topologies

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

Deliverable D2.1
Self-Organization Mechanisms for Economic Traffic Management

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:	Self-Organization Mechanisms for Economic Traffic Management
Type:	Public
Editor(s):	Simon Oechsner, Tobias Hoßfeld, Dirk Staehle
E-mail:	oechsner@informatik.uni-wuerzburg.de, hossfeld@informatik.uni-wuerzburg.de, dstaehle@informatik.uni-wuerzburg.de
Author(s):	George Stamoulis, Sergios Soursos, Ioanna Papafili, Jan Derkacz, Miroslav Kantor, Konstantin Pussep, Simon Oechsner, Tobias Hoßfeld, Dirk Staehle, Peter Racz, Spiros Spirou
Doc ID:	D2.1-v1.3.doc

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
V0.1	March 17, 2008	Simon Oechsner	First version, table of contents as discussed in Athens
V0.2	March 31, 2008	George Stamoulis	Structure changed slightly
V0.3	April 17, 2008	Simon Oechsner (editor)	Included bullet list contributions of WP2 partners
V0.4	April 22, 2008	Simon Oechsner (editor)	Inserted changes decided upon in the conf. call 18.04.
V1.0	May 30, 2008	Simon Oechsner (editor)	Integrated contributions from AGH, AUEB, DoCoMo, TID, TUD, UniWue, UZH from 23.05.2008
V1.2	June 17, 2008	Simon Oechsner (editor)	Included comments and improvements from AUEB, TID and AGH. Extended Executive summary and Conclusions. Final review version.
V1.3	June 30, 2008	Peter Racz, Spiros Spirou, Simon Oechsner (editor)	Included final review comments
V1.4	July 23, 2008	Burkhard Stiller	Final checks, ready for submission, ok

Table of Contents

1 Executive Summary

The objective of the SmoothIT project is to define, develop and test Economic Traffic Management (ETM) mechanisms to optimize the traffic impact of overlay applications on ISP and telecommunication operator networks based on a cooperation of network operators, overlay providers and application users. This document aims at providing an overview of self-organization in the context of the SmoothIT project, and how it can support ETM. It provides a definition and examples of self-organization for overlays. It also provides an overview of Self-Organization Mechanisms (SOMs) and their relation to the project goals.

In this deliverable, first results of Task 2.1 “Overview of Self-Organization Mechanisms for ETM” are discussed, which runs from Apr 08 – Dec 08. It also documents the milestone MS2.1 “Overview on Self-Organization Mechanisms for ETM”.

The goals of this document are 1) to give a first overview of SOMs, discussing their field of application, basic methods and results, 2) to describe the relationship between SOMs and ETM and 3) to provide a first evaluation of these mechanisms w.r.t. their usefulness for the project. This evaluation is based on the decision for a specific overlay application made in the parallel work package WP1, documented in deliverable D1.1.

The main outcome of this deliverable is:

- **Classification/description of current SOMs** able to implement ETM (Section 4)
- **Evaluation of SOMs** for further use with the application selected in WP1 (Section 4 and 5)
- Discussion of the **interworking of existing SOMs and ETM** (Section 6)

First, the most important terms are defined and related to each other in Section 2. A basic framework is described for the interaction of components like SOMs, metrics and ETM. SOMs are applied where choices must be made at a peer, and these choices are based on metrics. The aim of the project is to influence these choices so that ETM goals are achieved. The key to this is offering incentives to the peers or users, so that a decision that is good from the ETM perspective is also advantageous for the user.

After the theoretical foundation is laid, several overlay architectures are described and evaluated in Section 3. Examples for both search and distribution overlays are presented, covering Distributed Hash Tables (DHTs) as well as unstructured and hierarchical overlay networks. Focus is on widespread distribution systems using Multiple-Source Download (MSD), such as BitTorrent and eDonkey. Newer applications like Joost are mentioned, but due to the fact that they are closed-source not much detail can be given.

In the light of the decision made in the parallel work of WP1, namely the choice of a P2P (Peer-to-peer) Video-on-Demand (VoD) application for trials (see also D1.1, Section 7), the distribution overlays are concluded to be of more importance in the context of SmoothIT. Also, the higher amount of traffic they are producing makes them a better candidate for optimization.

After the variety of choices for overlay topologies has been illustrated, an overview is given on the mechanisms that can be applied to them in Section 4. As stated before, each of these mechanisms is based upon a local choice. The decision, *e.g.*, between two peers, is normally made based upon a metric that compares the choices, with different metrics being possible and used. They range from underlay metrics, such as the Round Trip Time (RTT) or provider-defined characteristics, to social metrics, *e.g.*, the reputation of a peer.

Since it is an important aim of the project to shape traffic according to ETM aspects, the described underlay metrics are rated as the most important. However, for them to be used in a SmoothIT solution, an information exchange must take place between the Internet Service Providers (ISP) in their role of an underlay provider and the peers making up the overlay. The basic principles governing this are described in D1.1, Section 6.

The actual mechanisms that utilize these metrics are classified in structure-forming and resource exchange mechanisms. While the latter govern which resources are exchanged between which peers, the former rule which overlay connections one peer establishes. However, most of the described SOMs that change the overlay structure are applicable only to structured search overlays, which are generally not an integral part of the envisioned VoD application. Nevertheless, it is the aim of this document to provide an overview without narrowing down the scope due to the decisions made in D1.1, therefore, these mechanisms are discussed for completeness' sake.

The resource exchange mechanisms on the other hand, such as the MSD or different chunk and peer selection strategies, are used in the applications that make up the application class selected in D1.1. Therefore, they are judged as important also in the next phases of the project. In particular, they influence the performance of a distribution overlay as perceived by the user, such as download times. The selection of data sources also influences which overlay connections are used for data transfer, making this decision process interesting for consideration in the project.

Since an overlay stores and manages resources, another aspect covered in this document is the (self-)organization of these resources. These SOMs aim at keeping resources, *e.g.*, data, available and also easily accessible in the overlay. While it is yet unclear whether the SmoothIT architecture will have a negative influence on these characteristics, no judgment can be given on their relevance for the project.

The information exchange between underlay and overlay is ideally bidirectional. To that end, also information about the overlay has to be gathered. A self-organizing way to do this is to use monitoring algorithms. Several are described in Section 4.3, focused on mechanisms applicable again in DHTs. However, many of the basic principles are general enough so that they may be used in an implementation of a SmoothIT architecture.

While the focus of this document is on self-organization, there are also several mechanisms that are not self-organizing per se, but that nevertheless allow an influence on overlays and therefore may support ETM. An overview on these is given in Section 5. Architectural options such as caches, proxies and crawlers are described, as well as the insertion of ISP-controlled peers into the overlay. Also, the effect of traffic re-routing and pricing is discussed. Most of these mechanisms require a strong intervention of the ISP. This raises concerns about network neutrality (see also Section 5 of D1.1) and also legality issues, making the presented options less desirable from an ISP's point of view.

Finally, Section 6 describes in more detail how SOMs can be used to support ETM. A specific example, based on the architectural discussions in WP1 and WP3, is explained and its implications pointed out. The approaches introducing ETM to overlays are classified by means of their transparency and illustrated by the example. Truly self-organizing approaches sacrifice transparency, because either the overlay provider or the end user (at least his P2P application client) has to adapt to the introduced changes. The most transparent options however again require a heavy investment of an ISP, with potentially less benefits for the other parties.

Also, the roles of the different parties involved in the process (end user, ISP and overlay provider) are considered. Aspects like security and the malicious use of information are introduced, but due to the scope of the deliverable not discussed in great detail.

The overview of SOMs for ETM presented in this document is of course by no means exhaustive. However, the most important mechanisms that could be of interest for evaluation in later project phases are presented, comprising a ‘tool-box’ useful for the implementation of ETM in overlays.

For this reason, this deliverable does not conduct a selection of certain mechanisms for later use. While tendencies for their usefulness can be estimated, at this stage a decision for a specific architecture would be premature.

Nevertheless, examples are given where necessary to create an understanding how SOMs can be used to implement Economic Traffic Management, using metrics and incentives.

The information contained in this deliverable will serve as the basis for the future work in Task 2.1, as well as providing input for other tasks in work package WP2. Also, the mechanism descriptions from this document serve as an information basis for Task 3.2 “Systems Architecture Design”.

2 Self-Organization and Economic Traffic Management

In this section, the terms self-organization and ETM in the context of the SmoothIT project are defined. To create a common understanding of these terms and of the interaction between them, their abstract characteristics are firstly described.

2.1 Self-Organization

While Self-Organization (SO) is known and defined differently in physics, chemistry, biology and a variety of other fields, a common denominator in all these definitions is that self-organization in a system increases its complexity without any external influence. In the context of the SmoothIT project, this means that a network structure evolves by local decisions made by the nodes participating in the network, without any higher authority directly intervening.

The term self-organization as we understand it in this project is defined on the level of overlay networks, i.e. logical networks above the physical network. By implication, the physical network is called the underlay. All SOMs (Self-organization Mechanism) described in this deliverable are implemented at the Application Layer, where an overlay generally provides a service or some functionality to an end-user application. These mechanisms aim at improving some characteristic of the supported application, such as faster response times or a higher degree of availability. Therefore, the term 'self-organization' may also be exchanged for 'self-optimization' in most of the mechanisms included in this document. However, for the sake of consistency, we will use the term self-organization throughout the rest of this deliverable.

A SOM is a concrete algorithm implemented at each peer forming the overlay. It makes the local decisions that, in interplay with the decisions made at other peers, achieve self-organization. In order to influence the overlay network, the SOMs presented in this document make some kind of choice, *e.g.*, between peers used as overlay neighbors. This choice is based on locally available data that is the input to the algorithm.

This data may be provided by other peers or by the underlay. The choice is made by applying a metric to this input. This metric provides semantics to the choice process by defining what makes one alternative 'better' than the other. To give a short example, using Round Trip Time (RTT) as a metric in a SOM would structure an overlay completely different than using the similarity in shared content as a criterion for selecting an overlay neighbor.

If the input for the SOM can not be provided by the peers themselves, the results of the SO depend on the quality of information available to the mechanism. A RTT measurement done by a peer, for example, may not be as reliable as similar information provided by the underlay itself.

2.2 Economic Traffic Management

Economic Traffic Management (ETM) is one of the key concepts of SmoothIT. Its main objective is to achieve the co-operation between the overlay and the underlay, resulting in traffic patterns that optimize the use of network resources according to some given criteria. This is attained by means of ETM mechanisms that are beneficial for all players involved. That is, such mechanisms promote mutual compatibility of the incentives for all players involved.

In particular, ETM employs mechanisms that are related to economic incentives of the users in the overlay. That is, they affect (overlay and underlay) decisions, leading to:

- a reduction of the economic cost incurred by the user and/or
- an improvement of the performance as perceived by the user.

At the same time, the way these incentive mechanisms operate and the state to which they lead the overlay is affected by information generated by the underlay and/or by policies employed therein. This way, the outcome of ETM is influenced by the underlay and its objectives. The objective of the ISP is to render this influence beneficial for himself as well.

The main toolkit of ETM is based on incentives mechanisms. The objective of such a mechanism is to shape the behavior of a participating agent by offering choices to him. The agent responds selfishly to the existence of the incentive mechanism and to choices he is offered. In particular, he performs such a selection so as to optimize his own objective function. That is, he adopts among the valid alternatives the one that optimizes this index. Usually, each choice represents a trade-off between: a) the utility for the agent by the outcome given his choice and b) the relevant cost for him. Of course, the "perfect" choice may not be permissible; *e.g.*, a high quality service at no cost!

To illustrate these, consider an ISP offering certain ADSL packages, namely different download rates at different charges. Some cases of the customers' objectives include the selection of: a) the highest rate that does not exceed a certain budget threshold, b) the lowest-cost package that exceeds a certain rate threshold, c) the package that represents the best value-for-money, i.e. the best trade-off between download rate and charge where both are considered as flexible.

A widely accepted objective function that quantifies such a trade-off is the net benefit, which equals the difference between the utility and the charge. In general, it is assumed that in order for a choice to be acceptable by an agent the resulting net benefit has to be non-negative. It should be noted however that the precise objective function may be private information to the agent. For example, the entity imposing the mechanism (in our case a service provider in either the overlay or the underlay) may know some possible types to which participating agents belong, the utility function associated with each type, and the distribution of users over the types. On the other hand, he may not know the precise type of each individual agent. Finally, in certain cases agents may not perform instantaneous optimization; on the contrary, they may also consider the longer-term implications of their selection and aim at optimization of a "long-term" objective. For example, in a p2p system, although instantly beneficial free-riding may not be long-term preferable for a peer; this can be avoided by means of a reputation-based mechanism encouraging contribution and rendering free-riding costly (see Section 4.1).

The information known by the provider is taken into account in order to optimize his own performance by the collective outcome. This can be seen as a problem of mechanism design; indeed according to [Pa05] "Mechanism design is the sub-field of microeconomics and game theory that considers how to implement good system-wide solutions to problems that involve multiple self-interested agents, each with private information about their preferences". (Similar definitions can be found elsewhere.) To summarize what applies to our case: the provider (or in general the entity setting the mechanism) imposes a mechanism, to which the participating agents respond selfishly and there arises an overall outcome; the mechanism should be such that the objective function of the provider is optimized. Of course, the provider should also take into account the fact that certain agents may not participate due to the mechanism. Consider again the example, of the ISP offering the ADSL packages; by doubling his prices, the ISP cannot expect doubling of his income, both because certain customers will reduce their rates, but also because others will not participate since they do not consider any of the packages as beneficial; *e.g.*, this applies to a customer who has a negative net benefit for all alternatives offered.

2.3 Self-Organization with Economic Traffic Management

The primary aim of the SmoothIT project is to implement SO mechanisms that enable ETM. Alternatively, better information may be provided for existing metrics in order to reach the same goal. This means that the overlay structure or behavior is influenced according to the goals of ETM. An example for this interaction will be given in Section 6.

SOMs will be one of the enablers of ETM under the SmoothIT approach. What makes SOMs useful to ETM is the fact that they run in the overlay; in fact most of the SOMs are already in place with the existing protocols. Of course, not all SOMs are appropriate for ETM. To see why this applies, we first classify the SOMs according to the selections they provide to the users in the overlay. In particular, the following possibilities apply:

1. SOMs offering no selections; *e.g.*, DHT-based content location in Chord.
2. SOMs offering to the user selections that are not based on immediate incentives: *e.g.*, the list of Kademlia-based “neighboring” peers, which, apart from preferring peers with a longer uptime, does not relate directly to any actual distance or other performance-related improvement.
3. Selections based on immediate application-layer incentives: *e.g.*, which chunk to download first in BitTorrent (“rarest fist replication”); this does relate to some performance-related improvement (namely maximization of the probability of the peer to be able to find all chunks of the file), which however is not related to the underlay conditions.
4. Selections based on immediate incentives that are related to the underlay: *e.g.*, bandwidth-based selection of peer to download from in BitTorrent.

It is mainly the last category of SOMs that can serve as enablers of ETM. In particular, the underlay provides agents (i.e. users) participating in such SOMs with the information employed in order to make the selections that the SOM actually prescribes; *e.g.*, RTT or other physical proximity metrics, for SOMs that prescribe that the selection of preferred peers is based on such metrics. This might for example be done by a) including such information to the tracker in BitTorrent, which will in turn provide it to the peers b) introducing a separate tracker in a BitTorrent like system that provides additional information about the network location of peers in a swarm, or c) having the underlay provide an interface for more reliable RTT measurements.

Furthermore, the selections made in the context of SOMs influence the traffic actually arising in the underlay, since they influence both the demand for traffic as well as the way it is routed. These interactions are depicted in Figure 1. Of course, the cases and the methods how the underlay can influence such SOMs for the benefit of all players involved are matters for study in SmoothIT.

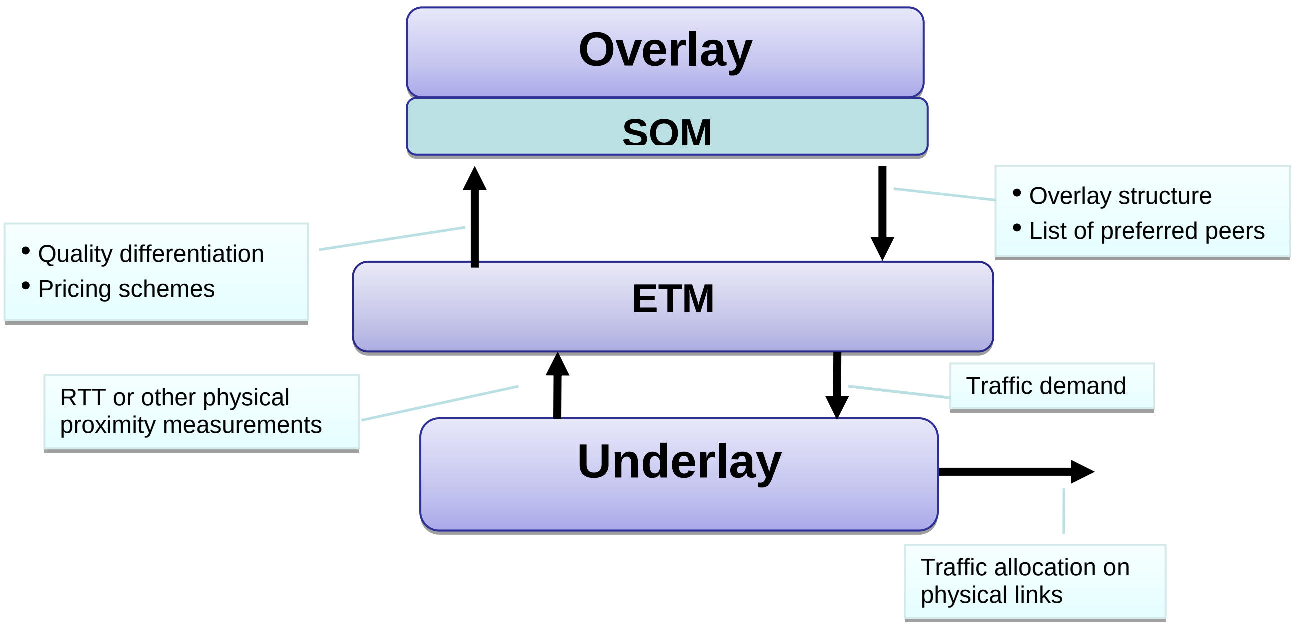

Overlay
SOM
• Overlay structure
• List of preferred peers
• Quality differentiation
• Pricing schemes
ETM
RTT or other physical proximity measurements
Traffic demand
Underlay
Traffic allocation on physical links

Figure 1: Interactions among underlay, ETM, SOMs and overlay.

Since most of the applications considered in the project and this deliverable are already implemented and working without the application of ETM, the latter should indeed provide extra benefits to the overlay layer, in order for these applications to adopt changes. This is to be achieved by offering incentives to all parties involved, *e.g.*, better QoS/QoE for end users and less cost for Internet Service Providers (ISP). A more detailed discussion on this topic can be found in Deliverable D1.1 [SI08]. It should also be noted that SOMs already run under the existing protocols. On the one hand, this imposes constraints to the way ETM can be realistically applied, but on the other hand does provide an opportunity and a challenge for SmoothIT.

Overlay applications play an important role in the SmoothIT project, and in the whole Internet market. An application might use more than one overlay network, *e.g.*, one search and one distribution overlay, and each overlay might be measured by several performance characteristics that should be optimized, *e.g.*, answer times, availability, throughput, etc. Therefore, we may not only consider one single SOM for a chosen application, but possibly several that work in parallel or even towards contradicting aims. The most widely known and deployed overlays are listed and described shortly in Section 3, while Section 4 provides an overview of SO metrics and mechanisms that may be used in a solution created in the SmoothIT project.

Another type of interaction between ETM and SOMs is plausible. In particular, the underlay can intervene actively in SOMs, *e.g.*, by introducing extra caching, or introducing “fake peers” just aiming to influence the SOM’s outcome for the actual peers. Clearly, such an interaction can be opted for all of the four categories of SOMs defined above. The intervention of the overlay in this case influences the outcome of the SOM and this in turn may influence the condition in the underlay. Again, it should be studied under which circumstances such interactions can serve the purpose of being mutually beneficial. To reflect this, Section 5 lists mechanisms that are not truly self-organizing, but that nevertheless may help to reach the goals of SmoothIT.

Furthermore, we briefly discuss pricing and its relation to SO and ETM. Pricing is one of the most prominent incentive-based mechanisms. Introduction of prices in the overlay can influence the structure and the operation of the overlay, and possibly lead to self-optimization, although this may not be its original intention. On the other hand, pricing in the underlay can have a direct impact on the traffic matrix and possibly on routing. If prices in the overlay depend on those in the underlay, then prices can serve the purpose of ETM.

Finally, in Section 6 we provide a more extensive analysis and some concrete examples to illustrate how SOMs and ETM are expected to interwork within the SmoothIT project.

3 Overlay Topologies

In this section, we describe some of the most common and best known overlay topologies. We discern search and distribution overlays. The latter is responsible for content exchange, while the former provides services to search for and locate specific content, i.e., files, documents, etc.

3.1 Search Overlays

Search overlays provide the functionality to locate resources in a distributed system. Any peer can issue a search query, which is then forwarded by the overlay and answered by the peers offering the requested resource. The actual usage of the resource, *e.g.*, the download of a file, is normally not handled by a search overlay. We will give a short overview of the search mechanisms and present different examples for them. It should be noted that the Distributed Hash Table (DHT) search networks without extensions support only exact-matching queries, while the Gnutella and eDonkey networks support keyword searches.

3.1.1 Resource Mediation Mechanisms

Methods used to search and locate resources in self-organized overlay networks are called resource mediation mechanisms. There are three general strategies of data retrieving for distributed systems [WGR05]:

- usage of central server: location of resources are stored by a central unit to which the queries are directed — best for simple and small overlays,
- search by flooding: location of resources are stored in the nodes actually storing the resources, they respond to flooded queries that are sent to all connected resources with a specified time to live — best for file sharing purposes,
- distributed indexing: the nodes are self-organized in some manner enabling specific routing for effective searching of resources.

Advantages and disadvantages of the three strategies are enumerated in Table .1

Searching and location of resources in self-organized overlays should be performed by some distributed and scalable mechanism. Distributed Hash Table (DHT) algorithms belong to the distributed indexing category and are currently considered as the best approach. DHTs apply the proactive strategy for data retrieval, because they structure the search space and then use the deterministic routing of queries.

When assessing resource mediation mechanisms the following metrics can be used [Tu05]:

- time needed to locate the resources,
- probability of finding certain resources,
- signaling overhead necessary to find certain resources.

Table 1. Comparison of advantages/disadvantages of three basic mediation methods

Strategy	Advantages	Disadvantages
Central server	Easiness with approaching the locating entity — low communication overhead; complex queries possible	Single point of failure; large memory consumption for one of entities
Flooding search	No need for proactive measures to maintain the overlay, low storage cost related to searching	Large communication overhead
Distributed indexing	Moderate (logarithmic) searching complexity, small communication overhead	Complex queries impossible

3.1.2 Distributed Hash Tables

A common characteristic of DHTs is the existence of a rule for the placement of data items, or, more commonly, the pointers to data items. A specific document or file has a key that is normally determined by the same hash function as the IDs (Identifier) of peers, or at least by a function with the same domain.

Each document can then be associated with exactly one peer, the peer with the ID that is closest to the documents key according to the keyspace metric (not to be confused with the SO metrics described later). This allows for a deterministic search, since for a requested document, just one well-defined node has to be located. Also, DHTs are designed with a certain structure in mind, enabling a routing process that locates this node efficiently.

As a consequence, DHTs can guarantee search success with a much higher probability than unstructured networks if a document is stored in the overlay. The DHTs of Chord, Kademlia, CAN and Pastry implement different structures, but follow the basic described principle.

- **Chord**

  Chord was published in 2001 by Stoica et al [SML+02]. It has since been investigated thoroughly in the academic world, although in practical applications it is not so popular. It organizes the participating peers in a ring structure, with each peer knowing a small number of successors on the ring. Additionally, shortcuts called ‘fingers’ are added to traverse the ring faster, resulting in a search in O(logN) hops, where N is the number of peers in the network.

- **Kademlia**

  Kademlia [MM02] is an efficient DHT implementation that imposes a less strict overlay structure on its peers, enabling it to use search traffic to enhance the stability of the network. It is based on the XOR metric defining the distance between peers, and uses buckets to manage the overlay connections of a peer. Each peer conceptually divides the whole ID space into regions by separating the ID space in the half containing the peer, and the rest. This is done recursively for the regions containing the peer, resulting in regions close to the peer's own ID being small, while the half of the ID space not including the peer is the largest region. From each region, *k* peers are chosen as overlay connection and are stored in a bucket for that region (the default setting for *k* = 20). This allows for a great degree of freedom for the neighbor choice in larger buckets.

Overlay stability is provided by using peers from which requests are received to replace offline peers in the corresponding bucket. However, if no peer has gone offline in a bucket, the new peer is discarded in favor of the older ones, exploiting the fact that a peer that has been online for a longer time is also more probable to stay online.

Kademlia also enables query routing in O(logN), with N being the number of peers in the network. It is used in implementations of several file-sharing networks, including eDonkey and certain BitTorrent clients.

- **CAN**

  The Content-Addressable Network (CAN) is a DHT that uses a d-dimensional torus as the basic structure of the overlay [RFH+01]. Here, each peer has a d-dimensional ID (possibly created by d different hash functions) that resembles its position in the coordinate space. The overlay IDs that a peer is responsible for is called its zone, and is the set of IDs that is closest to the peers ID in the Euclidean metric.

  During a churn event, i.e., when a peer joins or leaves the network, mechanisms have to re-allocate the zones in question. The amount of data a peer is responsible for depends also on the temporal development of the network.

  Routing works by forwarding messages in direction of the target coordinate, while it may be made more efficient by using more than one 'reality', i.e., more than one coordinate space, with peers having different IDs in the different realities. This allows for a choice for the next hop, a message is forwarded in the reality which offers the best conditions, *e.g.*, shortest RTT (Round Trip Time).

  The number of dimensions, *d*, may be used to tune the routing performance before deploying the network. For higher values of *d*, routing paths get shorter; however, the peer state becomes larger.

- **Pastry**

  Like Chord, Pastry [RD01] is based on a ring structure. It differs in the algorithm to determine overlay connections and in the routing process. Pastry uses prefix matching as the basis for its routing mechanism, trying to have a longer matched prefix between the current node and the target with each hop.

  The routing table is constructed by choosing peers with the right shared ID prefix for a given entry, with prefix lengths ranging from zero to the ID length. For each prefix, peers are chosen that cover the whole range for the next digit after the shared prefix. Apart from this routing table that is used in the first phase of the routing process, a neighbor table is used to have knowledge about the immediate surroundings of the local peer in the ID space. It serves in the second phase of the routing algorithm, when the message to be routed is close enough to the target node.

3.1.3 eDonkey

In the classic eDonkey network, documents are indexed using dedicated index servers that list the files available for sharing at the peers connected to them. These index servers form a separate layer, creating a small hierarchy with the peers themselves comprising the lower level. A peer that searches for a document queries the index server it is connected to, which then may forward this query also to the other servers, depending on the search parameters.

As an alternative to this also legally vulnerable concept, a Kademlia-based DHT has been added to the network, which serves as a distributed index. Both systems are run and can be used in parallel.

A detailed overview on the application characteristics and the variety of clients available for eDonkey can also be found in D1.1, Section 3.1.1.

3.1.4 BitTorrent

The search part of BitTorrent is actually not existent in the original protocol. BitTorrent relies on external sources to locate the .torrent files necessary for a peer to participate in a swarm. Currently, web search engines like piratebay.org are the most popular provider of such a service.

3.1.5 Gnutella v0.4 & v0.6

The original implementation of Gnutella (up to v0.4) connected the peers via a random graph. The content stored on each node is not advertised, *e.g.*, at an index server. Instead, a peer that is looking for a specific file floods this search request through the network in the hope of finding a peer that responds positively. As a result, much more peers are queried than necessary, and the message overhead incurred by the flooding mechanism makes this system rather inefficient and not scalable.

Search messages contain a TTL (Time to Live) value that is decreased with each overlay hop, in order to limit the search radius and duration. However, this also means that search success can not be guaranteed, even if the queried file is in the network.

To remedy these architectural drawbacks, Gnutella networks of newer versions (currently v0.6) form a two-tiered hierarchy with leaf nodes on the lower and so-called ultrapeers on the upper tier. Leaf nodes only have a small number of connections to ultrapeers and do not forward query messages. Ultrapeers maintain connections to their leaves and to other ultrapeers. Only nodes with sufficient resources and connectivity are eligible as ultrapeers. Each peer can decide by itself whether it may become an ultrapeer if these conditions are met, this decision also depends on the current network situation, *e.g.*, the number of available ultrapeers.

A leaf sends compressed information about the files it stores to its ultrapeer(s), which can then decide whether to forward a query to a leaf node or not based on a match between the query and the Query Routing table received from the peer. Queries are not forwarded if they have no chance of being answered successfully (in contrast to being only forwarded if they are sure to succeed).

3.1.6 Gnutella2

While the name Gnutella2 implies a version update from the 'original' Gnutella, it is more like a fork in the Gnutella development process. It also uses a hierarchical network structure, similar to that of newer versions of Gnutella, with several differences. The well connected peers on the upper level of the hierarchy are called hubs in Gnutella2, and share less connections with other hubs, but have much more leaves attached to them.

This is due to a difference in the search methodology. In Gnutella2, queries are not forwarded among the hubs, but a client that is searching retrieves a list of hubs first and then queries them consecutively.

3.1.7 Chordella

Chordella [ZDK06, ZDK08] is a hierarchical P2P system comprising two classes of nodes (*Figure 2*): high-power nodes (*superpeers*) are participants with substantial resources and a reliable network connection. They form a ring-shaped DHT overlay based on Chord. Any of these superpeers provide connectivity for the lower level nodes (*leafnodes*) using a single connection. Leafnodes do not need to provide a high speed Internet access nor do they require high computational power – they only need to fit weak requirements to participate in the network. This design is well-suited to be deployed on resource-restrained mobile devices.

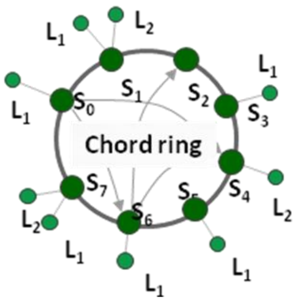


Figure 2. Hierarchical structure of the Chordella system

The superpeer client is able to act both as a superpeer and a leafnode. The leafnode client is able to connect to the network using a known superpeer, also via an air interface, such as WLAN (Wireless Local Area Network), GPRS (General Packet Radio Service) or UMTS (Universal Mobile Telecommunication System).

Apart from the basic DHT functions, the system also incorporates the following extensions for increased system performance:

A **Load Balancing Algorithm** [ZDK07] ensures approximately equal load levels of the participating superpeers. The current load level is piggybacked with messages that are exchanged between superpeers. It is then used to assign a newly joining leafnode to the least loaded superpeer. The algorithm has been shown to achieve a standard deviation of load levels across superpeers within a few percent.

The **Optimal Operation Point Selection Algorithm** dynamically adjusts the number of superpeers to a cost-optimal value. The algorithm realizes this based on local estimations of the global system parameters. According to the information gained in this way it promotes a leafnode to a superpeer if the experienced load is too high or demotes the superpeer to a leafnode if the load is too low.

An integrated **Caching Algorithm** improves the system performance and reliability by caching references to popular contents. An on-demand push algorithm populates the nodes' caches step-by-step backwards over the query path. References are not distributed along the whole path at once but only with increasing demand. This leads to shorter query paths and with that shorter query latency while at the same time minimizing the maintenance traffic needed for the caching algorithm.

3.2 Distribution Overlays

Distribution overlays are the logical networks that allow efficient access to resources, mainly data objects. While the actual data transfer is still handled via one-to-one connections, a distribution overlay allows for downloads from different peers, enhancing robustness and efficiency. We discern two general structures for these overlays, mesh-based and tree-based. Mesh-based overlays normally are more robust and easier to maintain, while tree-based overlays allow for the efficient use of application layer multicast.

3.2.1 Mesh-based Structure

- **eDonkey**

The eDonkey network uses Multiple-Source Download (MSD, see Section 4.1.4) to distribute content. To find sources for a file, a peer uses the search overlay. It then sends download queries to each source that is returned, where they are inserted into an upload queue. These connections constitute the distribution overlay of eDonkey.

The number of these connections is configurable per peer, depending on the resources available. At any time, only a few of these connections are actually used to transfer data.

eDonkey tries to prevent freeriding – the consumption of overlay resources without reciprocation - by granting an uploader credit points at the downloading peers, which are used to traverse the uploading queue at these peers quicker (see also Section 4.1.4).

- **BitTorrent**

A BitTorrent distribution overlay is formed for each file that is shared. It is formed by the tracker for that file, which is normally a server holding all information about the peers sharing this file and the status of their downloads. When a new peer joins the overlay, it contacts the tracker, which responds with a list of peers already participating in the overlay. The new peer then establishes connections with these other peers and waits for the possibility to download a chunk. The mechanisms for data exchange are described in Section 4.1.4. More technical details can also be found in D1.1, Section 3.1.2. Thus, the tracker is responsible for conducting the neighbor selection process and is therefore the best point to influence the structure of the network.

- **Gnutella**

Gnutella in its current form supports Multi-Source Download by using a protocol named Partial File Sharing Protocol (PFSP). To profit from this, more than one source must be known by a peer that wants to download a file.

To achieve this, information about sources is spread among the peers interested in a file and among the sources for that file. When a peer contacts a source for a download, the peer and the source exchange their respective data about other sources. For the downloading peer, these are sources the peer has found by searching and querying other sources. The source gathers information either by having downloaded the file earlier or by being contacted and informed by downloaders. This gossip-like protocol ensures that each peer knows a number of potential sources and that load can be distributed among them.

- **Joost**

The overlay built by the Joost application is used to deliver Video-on-Demand (VoD). Since Joost is a proprietary application, not many details are known about the structure of the overlay.

In general, peers that receive video data upload it in turn to other peers. It is assumed that the files are separated into chunks, just as in normal content distribution. However, since the peers are watching the video while downloading it, in order to guarantee the order, a buffer is used in the client to prevent degradation of the user's QoE.

Measurements have shown that one peer contacts quite a large number of other peers, implying that the general structure is a mesh [SI08]. Also, the overlay is server-assisted, i.e., content is provided by servers if the demand can not be satisfied by the peers. The share of content distributed by the servers depends probably on the network size, however, there is no information yet whether the Joost overlay is indeed scalable.

New peers can retrieve a list of available channels and probably also the addresses of already participating peers by contacting index servers, similar to the tracker concept of BitTorrent.

3.2.2 Tree-based Structure

- **Application Layer Multicast (ALM)**

Due to the lack of actually deployed IP-layer (Internet Protocol) multicast infrastructure, the alternative, Application Layer Multicast (ALM) is interesting for applications that need multicast functionality. ALM is a general concept covering tree-based overlay structures, and is therefore not one single type of network, but rather an architecture.

Numerous approaches for constructing multicast trees exist, an overview is provided, *e.g.*, in [HAS+07]. While not being as efficient as IP-layer multicast, ALM is deployed easier and can be tailored to the needs of a specific application.

A major differentiation criterion for ALM overlays are the metric that they aim to optimize, *e.g.*, end-user delay (low trees), bandwidth or efficiency (low outbound degree of nodes). Other classification criteria include:

- Deployment level, i.e., whether end systems are directly part of the multicast routing or whether proxies are employed
- Centralized or distributed management
- Direct construction of multicast tree (tree-first) or usage of a mesh to construct trees (mesh-first)

- Construction of source-specific or shared trees

Examples implementing ALM include:

- Splitstream

  SplitStream [CDK+03] also is a specific example for an ALM overlay, which is based on the Scribe group architecture, which is in turn based on Pastry. SplitStream tries to improve load distribution in a multicast group by splitting a stream into several so-called stripes. Each stripe is then distributed via a different multicast tree structure. However, all trees consist of the same nodes, with interior nodes in one tree being leaves in another. Thus, nodes can also control their utilized download bandwidth by subscribing only to a number of stripes.
- Narada
- NAIS
- ZIGZAG
- OMNI
- There are too many examples of ALM architectures to be properly described in this deliverable. However, all of the 27 examples referenced in [HAS+07] are to our knowledge not part of a popular application and, therefore, not widely deployed. References in [HAS+07] provide detailed descriptions of these architectures.

3.3 Relevance for SmoothIT

Since the application that will be in the focus for the trial phases has a mesh-based, P2P Video-on-demand (VoD) architecture based on BitTorrent [SI08], naturally all other architectures are of less interest in that respect. However, features from other distribution networks might still be included in the final application.

4 Self-Organization

Self-organization is addressed in this section according to the overlay structure seen and the resources utilized. Furthermore, monitoring aspects are discussed.

4.1 Self-Organization of Overlay Structure

Following a short introduction, the key metrics required to evaluate peers are summarized and categorized. Neighbor selection mechanisms are also discussed, which are followed by the explanation of resource exchange mechanisms.

4.1.1 Introduction

A self-organization of the overlay structure changes the logical links interconnecting peers. This is generally done at the peer level, where each peer makes local decisions, *e.g.*, on the peers it selects as neighbors, which peers to upload to, etc. These decisions are made based on locally available information about the candidate peers. The outcome of all these local changes in their sum affects the complete overlay structure.

A selection among two or more candidates is made based on a ranking of these peers according to a metric. We will describe some possible metrics first before listing several SOMs that utilize them. For the SOMs, we discern between neighbor selection and resource exchange mechanisms. The former govern which peers are directly known by a local peer, i.e., to which peers an overlay connection is maintained. The latter are applied for utilizing these connections, especially in the context of content distribution. Specifically, they govern with which neighbors data is exchanged and what part of that data.

4.1.2 Metrics for Peer Evaluation

Typical overlay-based applications have some degree of freedom in choosing their neighbors. For example, in the Kademlia DHT there are often several alternative peers which can be put in a certain bucket. Similarly, in a content distribution overlay such as BitTorrent, a peer can decide which potential contact to ask for a specific file or chunk and which requesting peer is to be served.

Many of the SOMs presented in this document are based on influencing this choice. Peers or chunks are selected with a certain goal in mind, which makes one peer a better choice than another. This ranking of peers for a given purpose, *e.g.*, uploading a chunk, is made by applying a metric.

The simplest ‘metric’ that can be applied is a random choice between the candidates. To make a more informed decision, different attributes can be considered. They can correspond to either the underlying network, where a number of technical metrics are possible, semantic or social factors, such as fairness, similarity or social association between peers, or geographical distances, possibly because of different legal regulations etc. Another very important metric is the underlying network’s cost, which is not considered by the most overlay applications. This cost is in the focus of SmoothIT.

The applicability of the metrics depends on the overlay type and goal, as *e.g.*, search overlays are interested in other performance indicators (such as RTT) than Content Distribution Overlays (often throughput, in case of VoD or live streaming also RTT). In the following, we discuss the most important metrics in the context of SmoothIT, i.e. their relevance for Internet Service Providers, users and overlay providers.

Load-balancing metrics

The main target of load-balancing metrics is to distribute the load uniformly across peers in the network, and therefore to achieve fairness and, probably, a better performance. Given a set of candidate contacts the peer would select in such a manner that each peer would receive roughly the same number of requests: the easiest way to do so is to select random peers. Another possibility is to iterate peers according to the round-robin scheduling. In a fully decentralized P2P system, such as Gnutella, this decision is made based on a local knowledge of the peer and may not result in the desired fairness, as some peers might have a higher number of connections than others. In contrast, in a hybrid system the centralized components may take care that each peer receives roughly the same number of requests. While such a metric is easy to implement, it does not take the heterogeneity of peers into account. While it may be fair in principle to let each peer serve the same amount of requests, in practice there may be peers that have a lower connectivity or less computing resources, *e.g.*, mobile phones. In this case a mobile would be overloaded, while a PC (Personal Computer) connected via DSL (Digital Subscriber Line) would handle the same number of requests with no problem at all.

On the other hand, basic load-balancing is necessary in any overlay in order to prevent traffic 'hot-spots' that may occur if the overlay is only optimized with a narrow definition of efficiency in mind.

Underlay metrics

To reflect characteristics of the physical network, peers may rate candidate contacts according to their relationship in the underlying network, i.e. typically the Internet. Possible metrics include the Round Trip Time (either measured or predicted), number of underlay hops (measured *e.g.*, via traceroute), topological proximity (obtained via the length of common IP prefix or external IP lookup services such as MaxMind [MM08]) and the available bandwidth between the peers.

This kind of metric tries to level the information asymmetry between overlay and underlay, i.e., the fact that the overlay knows little or even nothing about the structure of the underlying physical network. This lack of information leads to inefficient overlay connections, both in the sense of end user QoE and resource consumption in the overlay. As an example, one overlay hop may span several intercontinental links, leading to resource wasting in the physical network.

As seen from the examples the required information for these metrics can be obtained by different means:

1. local measurements between peers,
2. overlay-wide measurements, or
3. external public services.

While local measurements are closest to the fully distributed P2P idea, there are two major drawbacks to this approach. Not all network characteristics a peer might want to know about are measurable. Examples would be link bandwidths or inter-AS routing policies. Also, the characteristics that can be measured may only be quantified with certain accuracy. RTT measurements, *e.g.*, might be more accurate than the physical hop count between two peers, but may fluctuate over time.

These problems are acerbated if the measurements are done overlay-wide. In this case, the delay introduced by collecting and distributing measurements may lead to outdated results.

The third option may be able to provide the best information, if the network provider is offering the external service. In the context of SmoothIT, it will be of interest how this better information about the physical network can contribute to a higher efficiency of the overlay and at the same time to a provider-friendly structure.

In the case of external services the information may be offered by an ISP (as proposed for Deutsche Telekom oracle or P4P [PP08]) in order to steer the overlay construction.

The following main metrics for the underlay are identified:

- **RTT**

  Being the primary example metric for most mechanisms, the RTT (Round Trip Time) tries to capture the underlay network distance between two peers. Especially for applications with timing constraints, it might be beneficial to maintain connections primarily to peers that only have a small delay. It is included, *e.g.*, in the specification of Pastry [RD01]. The positive influence of a short RTT on the download times of peers is described in, *e.g.*, [EHB+07].

  However, while being a good measure, the RTT does not necessarily provide an exact picture of network conditions, especially since it may vary over time. RTTs between peers can be either measured directly (via pings) or estimated based on virtual coordinates, *e.g.*, such as Vivaldi [DCK+04].

- **Hops, topological proximity**

  A metric similar to the RTT in spirit is the physical hop count between two peers. While being more coarse-grained, it is more stable over time. It does however not capture link lengths or traffic conditions. Physical hop counts are also difficult to determine locally by the peers, since not all providers allow their routers to react to the ICMP (Internet Control Message Protocol) messages used to measure hops.

- **Provider dependent information**

  In today’s networks, an important cost supported by ISPs is the interconnection cost that will depend on the type of the interconnection agreement and on the amount of traffic sent and/or received. In the current Internet backbone (and therefore in the interconnection links), the most important contributor is P2P file sharing traffic.

  For each IP the ISP can get the affiliation information and the costs associated to send and or receive traffic in the following way:

  1. The ISP can check from the BGP (Border Gateway Protocol) information (usually available in BGP route reflectors) the AS (Autonomous System) to which the IP belongs and the set of AS’s that must be crossed. This information is maintained in BGP as the network_prefix and mask, the ASPath and the next_hop fields in the routing tables.
  2. According to next AS, the ISP can infer the costs associated to this peer, depending on the type of agreement that could be: peering (in this case, the ISP

would be interested in maintaining the symmetry in the traffic) and transit (in this case the cost associated to this peer)

The routing policies are usually set in such a way that the ISPs maintain interconnection peering agreements with all those ISPs that are interested on terminating connections and that are physically reachable without high deployment costs. When a peering agreement to a given destination ISP is not available, the ISP has to configure the routing policies in such a way that all the traffic that must be sent to such destination, traverses a Tier-1 provider (that acts as a hub) which has a transit interconnection agreement.

It is important to highlight that the ISP will be interested on optimizing the costs associated to traffic management but able to also improve the overlay performance (*e.g.*, better QoS, faster downloads for P2P file sharing applications). Therefore, SmoothIT must provide a solution able to improve the performance of the overlay applications and that should reduce the costs associated to the overlay applications.

- **Geographical distance**

  Geographical distance between two peers is computed using the peers' geo-coordinates, such as Universal Transverse Mercator (UTM). This approach is used in, *e.g.*, [HOT+06]. This metric only indirectly maps the physical network. It uses the assumption that peers with physically close positions are also well connected in the network. Since this is not true in many cases (*e.g.*, the closest peers could be attached to different ISPs), a potential mismatch between the proximity returned by this metric and the actual network conditions occurs, leading to the same inefficiency problems as described above.

  On the other hand, this method has the advantage that geo-coordinates can be measured easily with a sufficient accuracy, provided a peer has the technology to do so. For example, many current mobile user devices include a built-in GPS receiver. However, if this can not be done automatically, *e.g.*, for end user PCs, the configuration effort of the peers increases.

- **Link Bandwidth (last mile)**

  For content delivery overlays the peer's bandwidth has an important impact on the download performance. Peer's upload bandwidth being typically much smaller than the download bandwidth can be used to prioritize peers. Due to the fact that the actual available capacity may continuously change it is much simpler to use the maximum available capacity of a peer. An additional reason to do so is that the maximum bandwidth is much easier to determine than the available bandwidth. Most overlays consider the available bandwidth only indirectly by rejecting the request if the transfer rate is too low.

  Search overlays are much less bandwidth sensitive, however, in heterogeneous environments mobile phones and other weak devices can be disburdened by taking the access bandwidth into account.

  In P2P file sharing systems, due to the current tit-for-tat mechanisms, users with similar bandwidth capacities associate themselves. Therefore, taking into account current mechanisms, in order to get higher throughputs in P2P file sharing, it is mandatory to improve the access speed (taking special importance the usual asymmetry between uplink and downlink bandwidths); in this way, the P2P application will be allowed to use more bandwidth to share content and, therefore, to get content from peers with faster network access.

In order to correctly evaluate the mechanisms defined in SmoothIT, the selection of a realistic scenario where several access technologies (with different speeds) deployed in different Autonomous Systems is mandatory. In such a way that the implementation of any ETM must benefit to all the ISPs and their end users when they want to use any overlay application. We should avoid, *e.g*., the implementation of a solution just based on locality that could only benefit the ISPs with higher bandwidths in their access and could lead to a decrease of the download time for users belonging to ISPs with lower bandwidths in their last mile.

Social metrics

Another possibility viable for certain applications is to rate candidate contacts according to an application specific metric such as peer reputation, common interests, similarity of shared content, or even social interrelationships between users. In general these metrics are completely decoupled from the underlay metrics and therefore, often, have a negative impact on the underlay proximity.

Because the reason of applying such a metric is typically not the (short-term) performance of the system but rather application-specific they cannot be influenced by ETMs directly. Instead they might result in a set of candidates with similar rating which can then be further filtered according to some other metric.

- **User interests (semantics), clustering**

  This metric defines the distance between peers based on user interests. Two peers are defined as being close if they share the same interests. An example for this would be the search for similar files in a file sharing network [HKF+04]. A user that is primarily interested in rock songs is semantically closer to a peer that also looks for rock songs than one that is interested in techno music.

  Being better connected to peers with similar interests is advantageous in terms of search success if an unstructured network is used. The reason for this is that the probability to search for content that can be provided by semantically close neighbors is higher.

- **Reputation**

  Introduction - The issue of Free-Riding

  The appearance of new forms of peer-to-peer applications, such as eMule and BitTorrent, holds promise for the continuous emergence of fully distributed information sharing systems. These systems, inspired by Napster back in 2000, allow users worldwide access and provision of information while enjoying a level of privacy not possible in the present client-server architecture of the Web. While a considerable attention has been placed on the issue of free access to content (*e.g*., music) and on the violation of copyright laws through these systems, there remains an additional problem of securing enough cooperation in such large and anonymous systems so they become truly useful. Since users are not monitored as to who makes their files available to the rest of the network (produce) or downloads remote files (consume), nor are statistics maintained, it is possible that as the user community in such networks gets large, users will stop producing and consume only. This *free-riding* behavior is the result of a social dilemma that all users of such systems confront, even though they may not be aware of its existence. In a general social dilemma, a group of people attempts to utilize a common good in the absence of central authority. In the case of a system like Gnutella, one common good is the provision of a very large library of files, music and other documents to the user community. Another might be the shared bandwidth in the system. The dilemma for each individual is then to either contribute to the common good, or to free ride on the work of others. Since files on Gnutella are treated like a public good and the users are not “charged” (not even virtually) in proportion to their use, it appears rational for people to download music files without contributing by making their own files accessible to other users. Because every individual can reason this way and free ride on the efforts of others, the whole system's performance can degrade considerably, which makes everyone worse off. This is the well-known tragedy of the digital commons [Ha68]. The second problem caused by free-riding is to create vulnerabilities for a system in which there is risk to individuals. If only a few individuals contribute to the public good, these few peers effectively act as centralized servers. Users in such an environment thus become vulnerable to lawsuits, denial of service attacks, and potential loss of privacy. This is relevant in light of the fact that systems such as Gnutella, Napster, and FreeNet are depicted as a means for individuals to rally around certain community goals and to "hide" among others with the same goals. These may include providing a forum for free speech, changing copyright laws, and providing privacy to individuals.

Therefore, free-riding is an important issue in the peer-to-peer paradigm for exchanging services, such as digitized content. The value of such a service for the client peer depends on the performance level of the peer providing it, which may be unacceptably low. The reason for this can be either the peer's hidden type (i.e., his strategy on service provision) or his hidden quality (i.e., his actual ability to provide the service). Reputation can be a proper means of revealing low-performing peers in peer-to-peer electronic environments (as well as in markets), if it is defined appropriately and calculated accurately; see [De03], [JHF03]. Clearly, reputation breaks the information asymmetry between the peer offered the service and that providing it. Low-performance can be selected by the latter as a means of free-riding. In this sense, reputation also contributes to alleviating the problem of free-riding. Some of the mechanisms reviewed in Subsection 4.1.4 serve this purpose, either this way, or by encouraging contribution of content to the system.

Definitions and Calculation of Reputation

Generally, modeling reputation implies that agents are involved in repeated interactions and that they are uncertain about specific relevant properties of their opponents. For example, game theory introduces the concepts of repeated games and Bayesian types for this purpose. The interaction may be structured in such a way that each agent’s utility is influenced by the unknown property of the other agent with whom it interacts, or it can be that only one agent is interested in discovering the other agent’s unknown property. Now, the key is that each interaction outcome reveals some new information about the unknown property of a considered agent and thus it makes sense that it is made publicly available so that the future consumers of that agent’ services can tune their behavior to reduce the risks and shape their interactions in an appropriate way. All outcomes of interactions of a specific agent providing a service, either in a raw or an aggregated form constitute that agent's reputation.

Each peer may either succeed or fail in offering a service to another peer; the likelihood of each *outcome* depends on several factors, such as the quality of his network resources or of his content. Actually, only the outcome of each transaction is of interest to a peer to whom a service was provided, rather than the hidden reason for this outcome. After observing the outcome of his transaction, a client peer *rates* the providing one for his performance. As already mentioned, all ratings received by a specific agent providing a service, either in a raw or an aggregated form, constitute that agent's reputation. Assume for the moment that peers report truthfully their evaluations for the performance of others. Binary rating (i.e., success vs. failure) is known to be adequate for calculating a reputation value that is representative of the expected outcome of a transaction with a specific peer [De03]. The *aggregation* of all ratings that a specific peer got for the services he provided in a single reputation value is also important for practical reasons (*e.g*., the storage overhead is reduced). According to [De03], *Bayes'* rule is an efficient aggregation function, which can be applied if peers are classified in types, each having a fixed probability for successfully providing services, and there are defined initial beliefs on these probabilities and on the proportions of the types in the peer-to-peer system. In particular, reputation of each peer then expresses the *a posteriori* belief that he belongs to a certain type given the history of his service provisions. In particular, given that there is a fraction γ of peers offering services of high performance with probability p_H and that the other peers offer services of high performance with probability p_L with $p_L < p_H$, then the belief that a certain peer belongs to the high performing type after a successfully (resp. unsuccessfully) offered service is given by the first (resp. second) formula below:

$$P(high \mid success) = \frac{\gamma \cdot p_H}{(1-\gamma)p_L + \gamma \cdot p_H}$$

$$P(high \mid failure) = \frac{\gamma \cdot (1-p_H)}{(1-\gamma)(1-p_L) + \gamma \cdot (1-p_H)}$$

If the distribution of peer types in the peer-to-peer system is renewed dynamically and/or peers follow dynamic strategies over time (thus altering their probability of success) rather than fixed ones, then Beta aggregation is more appropriate, as it does not depend on any prior beliefs for the distribution of peer types in the system. According to this approach, each peer's reputation equals the fraction of the "weighted number" of his successful service provisions over the "weighted total number" of his service provisions, with the weight of each service provision being a negative exponential function of the elapsed time. This approach is called Beta aggregation, and was introduced in [JHF03]. Specifically, it is described by the formulas below.

$$\boldsymbol{R} = \frac{\boldsymbol{s'}}{\boldsymbol{t'}}$$

where $s' = s \cdot d + \mathbf{1}(success)$ and $t' = t \cdot d + 1$

R is the new reputation value of the peer; s is the sum of as many unit as the number of the previous successes, with each unit term being discounted exponentially in the time distance from the present; t is the discounted number of services he provided; s´, t´ are the updated values of s, t after a new service provision by the peer; 1(.) is the indicator function. Finally, d is a discount factor denoting the relative importance of the past history of ratings over against the recent one.

Other metrics to be considered

Taking into account the current efforts to secure any Internet service, **security** could become an important issue to be taken into account to set up an overlay. In this context, in a P2P network, peers could be more interested in downloading the chunks from secured sources. Therefore, the need to evaluate the security level of a peer could arise in the near future.

In this context, some ISPs are also offering services to secure the home devices such as anti-virus software or securing monitor agents; so, the ISPs could have information about the peers that have, *e.g.*, installed the anti-virus or installed the recent Windows patches. In this way, we could order the peers according to its security level.

Combined metrics

As can be seen from the description of different metrics, it is often undesirable to use only one metric for candidate selection. The reason is that an overlay might have concurrent goals to base the decision upon. For example, in a BitTorrent overlay the content sources are selected based on the available upload bandwidth but at the same time a tit-for-tat mechanism is applied to prevent free-riding.

Relevance for SmoothIT

Since it is the aim of the SmoothIT project to direct traffic to specific parts of a network, i.e., to keep traffic from leaving the local ISP's network, the underlay metrics described may play a larger role than the others when selecting neighbors for overlay connections. Depending on the performance of the application, other metrics might however still be included in the selection process.

4.1.3 Neighbor Selection Mechanisms

An overlay can be seen as a graph with overlay nodes being vertices of this graph and the connections between nodes being edges of the graph. In this section we consider only long-living connections: *e.g.*, in Kademlia [MM02] each node has a set of connections to other nodes whose contacts are stored in the routing table. As long as both nodes are alive this unidirectional connection persists, unless a better contact is found. We call such contacts *neighbors*. During the (iterative) routing process a node may gain knowledge about other contacts and even exchange data with them, but unless they are not stored in the routing table they do not become neighbors. Note that this neighborhood does not have to be related to any neighborhood relation from the real world, such as geographical or underlay related proximity.

The choice of suitable neighbors is done by the *neighbor selection mechanisms.* Such a mechanism is often overlay-specific, depending on what an overlay tries to optimize: (end-to-end) delay, throughput, loss etc. For example, a search overlay such as a DHT or Gnutella-like overlay will typically try to optimize the delay (additionally to the cohesion of the graph and load distribution) while a BitTorrent-like system will be much more interested in high throughput in order to distribute large number of bytes as fast as possible.

Additionally to these overlay-specific goals underlay-specific goals such as optimal usage of network resources can be taken into account. As discussed in Section 4.1.2, making nodes from different autonomous systems to be neighbors in the overlay can result in significant amount of inter-domain traffic which can be reduced if nodes try to select neighbors from the same autonomous system [AFS07].This ISP-related goal may correlate with overlay-specific goals such as delay optimization, but it does not have to.

In this context we must again distinguish between search overlays and content distribution overlays. Typically, neighbor selection is very important in a search overlay in order to assure short answer times and high success rates per query. The neighbor relationships are very pronounced here and only change when existing peers go offline or new peers join. This is very different in distribution overlays where nodes are only connected as long as they exchange some data. So for distribution overlays the mechanisms discussed in the next section (Resource Exchange Mechanisms) might have higher influence on the overlay performance.

Proximity neighbor selection

Proximity neighbor selection [CDH+02, GGG+03] is a mechanism that influences the routing table of a peer, as described above. A prerequisite for this is a choice in the peers that are included in the local peers' routing table. Then, from the possible choices, the best peer according to a metric is selected as the routing table entry.

Since this action is taken whenever a position in the routing table is to be filled, this mechanism influences the overlay layout mainly during the startup phase of a peer and during updates of a routing table resulting from churn.

The effectiveness of this method may depend on the degree of freedom for the choice of peers. If only a very small number of peers may be selected (*e.g.*, in an DHT), the potential gain could be smaller than *e.g.*, in an unstructured overlay (where potentially every peer may be selected).

Proximity routing

Related to proximity neighbor selection, proximity routing [CDH+02, GGG+03] also favors peers that are close to the local peer in terms of a metric. However, instead of making this choice during the creation and maintenance of the routing table, the alternatives for a next hop are included as entries. Only when a message has to be forwarded, the current best choice from these alternatives is taken as the next hop.

The selection of the next hop during a message forwarding can also take into account short term fluctuations in network conditions. However, this is paid for by the larger routing table and a potentially higher routing complexity. Also, the optimization potential depends on the choices available.

Geographic layout

Another approach similar to proximity routing and proximity neighbor selection is the geographic layout [CDH+02, GGG+03]. Here, the aim is to structure the overlay in a way that preserves neighbor relationships from the physical network, i.e., peers that are close in the physical network according to a metric are ideally also neighbors in the overlay. This can be done by adapting the ID generating process of the peers, so that locality information pertaining to the metric is included.

The goal of this method is to create a mapping between overlay and underlay, thereby trying to limit the overhead introduced by single overlay hops that span several long physical links.

This mechanism is mostly applicable to DHTs, since they create relations based on node IDs.

Supernode selection strategies

The term supernodes denotes peers that play a special role in the network, somewhat violating the pure P2P-approach. Examples are the ultrapeers or hubs from Gnutella, or the supernodes in the Skype network. They are normally selected from the complete set of peers.

Supernodes are employed to reduce load on ‘normal’ peers. In the example of Gnutella, they handle most of the query traffic, while in Skype they are also used to establish indirect voice connections between two peers that can not connect directly to each other, *e.g.*, due to firewalls or NATs.

As a consequence, not every peer is useful to be selected as a supernode (or to promote itself to that status). Normally, peers are useful as supernodes if they can provide enough resources, *e.g.*, bandwidth and processing power, and if they are easily reachable by all peers. If these conditions are met, a peer might become a supernode if not enough supernodes exist in the network.

Relevance for SmoothIT

Due to the fact that proximity neighbor selection, proximity routing and geographic layout are to be seen in the context of DHTs, which as search overlays do not directly play a role in a distribution network for VoD streaming, these mechanisms are not currently of major interest. The concept of super-peers might however still play a role if different peer capabilities are considered.

4.1.4 Resource Exchange Mechanisms

Multi-source download

Under Multi-Source download (MSD), we understand here the possibility to download parts of the same file from different sources, i.e., peers, in parallel. There is also another interpretation, which only defines MSD as having more than one source for a file, so that when a download from one source fails or is interrupted, it can be continued by using another source. However, a download is only started from one source at a time in this context.

All currently popular distribution overlays use multi-source download to exchange data. Peers are not limited to one connection at a time, but may receive data of the same file from several peers at the same time. Similarly, a peer may upload data to more than one peer at the same time.

To enable MSD, files are usually segmented into smaller parts, which can be shared by a peer even if the complete file has not yet been downloaded. This also greatly speeds up the spread of especially larger files, since peers can start to upload data much sooner. The file segments are commonly called chunks, which are in many cases partitioned again.

Cooperation strategies (chunk & peer selection, incentives)

In the context of resource exchange mechanisms, two major selection algorithms are used that can together be described as cooperation strategy. To explain these algorithms, we consider a peer X that wants to download content. Since most distribution overlays use multi-source download, it signals its download wish to several other peers. The choice of peers that are asked for content is one part of the peer selection process: the peer selection at the downloading peer.

A peer Y receiving X's download request queues it locally with all the other download requests. When resources for a new upload at Y are freed, then Y has to decide which peer it wants to upload data to. This is the peer selection at the uploading peer.

Finally, when peer X was selected for an upload slot at peer Y, it has to be decided which part of the file, i.e., which chunk, will be transferred. Of course, only chunks that Y has stored can be uploaded, and only chunks that X is missing are useful. The selection takes place from the overlap of these chunk sets.

Peer selection strategies

- Tit-for-tat

  The distribution of a content file in BitTorrent proceeds by peers exchanging data with each other until every peer owns a complete copy of the file. Each peer p tries to maximize its own utility, *e.g.*, its own download rate and ultimately its overall downloading time. When two peers experience poor downloading rates they are often able to optimize their downloading rates by uploading to each other. As a result they achieve pareto efficiency, that is they reach a situation where they can make no other exchange and receive better utility.

  BitTorrent's choking algorithms try to achieve pareto efficiency by assuring that peers reciprocate by uploading to peers that upload to them or – the other side of the coin – that peers refuse to upload to peers that do not upload to them. Disallowing downloading from peers is called choking, while allowing is called unchoking, *e.g.*, peer p is choked at peer q if it is in the neighbor list of peer q, but not currently downloading anything. Peer p normally becomes unchoked when peer q experiences a good download rate from peer p. The choking algorithm tries to ensure that when a peer q uploads data with a certain rate to peer p, it gets the same amount back from p (although getting a different part of the common file shared). Thus, some measure of fairness is guaranteed. So-called 'freeriding' is made more difficult by this mechanism. A prerequisite for this method to work is that both peers have content to offer that the other peer needs. This strategy is a variant of the tit-for-tat strategy that was used to play repeated prisoner's dilemma game.

Practically, each BitTorrent peer p always unchokes a fixed number of its peers. By default all connections are choked. The default number of peers that can be unchoked by p at a time is five. The peer p decides once every ten seconds a set of four of its peers to unchoke based on the current downloading rates that p experiences by its peers. Each peer keeps track of the downloading rates it receives by all of its peers. Calculating the current downloading rate is difficult and it is based on a rolling twenty-second average, i.e., only the last twenty seconds rates are taken into account. This policy deters freeriding, *e.g.*, if a peer p downloads from a peer q but being a freerider p refuses to upload to q, then the next time that q will re-determinate the list of its unchoked peers, q will choke p because of the poor (or zero) downloading rate offered thereby.

Additionally, a random peer is unchoked optimistically in order to start new resource exchanges, allowing peers with little content to gain more data to exchange. This is the fifth peer being unchoked *regardless* of the current downloading rate. This policy is called optimistic unchoke because it allows peers to download data even if they have not proven that they will upload something in return and it is applied every thirty seconds. This happens basically for two reasons: First, in order to avoid the starvation of new peers in a swarm (which have nothing to offer to other peers initially) and second, for old peers to discover new potentially better connections. For instance, suppose peer q is a newcomer and has asked to download some pieces from peer p. Peer p not having adequate info about q's behavior may not upload to peer q which could provide a higher downloading rate to p than any other of p's unchoked peers. By BitTorrent allowing p to unchoke a peer regardless of the downloading rate, p could unchoke q, which in the next twenty seconds will in turn unchoke p. As a result p has discovered a better new connection. Optimistic unchoke corresponds to always play 'cooperate' on the first round of repeated prisoner's dilemma game.

BitTorrent's choking algorithm implies that a peer p rewards peers that uploaded to p a bit earlier. The purpose of a choking algorithm is to utilize all available resources, *e.g.*, upload capacity of all peers, to provide reasonably consistent download rates for all peers, to "punish" freeriders, *e.g.*, peers only downloading and not uploading, and on the other hand reward peers contributing to the system's overall efficiency. The choking algorithm is not technically part of the BitTorrent protocol, but is an incentive mechanism that assures good overall performance.

- Credit-point systems

  The credit point system used by the eMule client (a client used to connect to the eDonkey network) builds up points at the peers that content is uploaded to. If peer A uploads some amount of data to peer B, B adds credit points for A locally. These credit points allow A to spend less time in the upload queue of B if A wants to download something from this peer. Since all credit points benefiting A are stored on remote peers, it is assumed that no counterfeiting of credit points is possible.

  An interesting difference between the tit-for-tat mechanism and this credit point system is the fact that credit points accumulated by uploading one or more files can be used to download *other* files quicker, whereas tit-for-tat tries to impose fairness in the sharing of the *same* file.

- Reputation based mechanisms

  Several reputation-based mechanisms have been proposed and evaluated, mostly for peer-to-peer systems. Except for improving the performance experienced by the various peers in their transactions with others, an important issue also dealt with in several of these works is as follows: When reputation of a peer is calculated on the basis of ratings of other peers (rather than just on personal ratings of the peer to employ the reputation value), then these ratings cannot be assumed to be truthful, unless the mechanism also offers incentives for submission of truthful ratings.

  In the following, we overview most of the relevant works:

  - Ma et al. propose in [MLL+04] a bidding mechanism for the proportionally fair allocation of the resources of a providing peer to the contribution values of the requesting peers in the peer-to-peer system. Providing peers are selected at random.
  - Papaioannou and Stamoulis prove in [PS06] that employing a reputation metric is not adequate, unless it is complemented with the use of appropriate policies. They also showed the effectiveness of applying a pair of policies for provider selection and contention resolution. The same authors also present in [PS05] a mechanism promoting truthful submission of ratings in the calculation of reputation.
  - In [HW04] Hwang and Lee investigate the evolution of different species of peers that follow different strategies regarding the offering content while reputation metrics for usage and service are employed. It is shown that the evolution of the various species is done in favor of the high-performing peers.
  - Schillo et al. deal in [SFR00] separately with the behavior and credibility of peers (referred to as agents therein) using the so-called disclosed prisoners' dilemma game with partner selection. Performance (due to strategic behavior) and credibility of other agents are updated by an agent's own observations, while testimonies of witness agents are used for partner selection.
  - In [DCP+03] Damiani et al. extend Gnutella protocol to calculate performance and credibility of other peers based on a peer's own experience and votes from witnesses. This approach (referred to as P2Prep) is similar to the one of [SFR00] in several aspects.
  - Performance (due to strategic behavior) and credibility are addressed by Yu and Singh [YS02]. However, this approach has no explicit mechanism for assessing the credibility of the witnesses; this issue is dealt together with a trust metric regarding behavior, which is determined by direct observations or by asking witnesses. Therefore, it is possible for an adversary peer to maintain a good reputation by performing high quality services and send false feedback for its competitors or his colleagues.
  - This argument also applies to the approach of Kamvar et al. [KSG03] where, a global reputation metric regarding performance of each peer is calculated. To this end, each peer's local beliefs (based on observations) on the performance of other peers are weighted by the others' beliefs on his own performance.

- Aberer and Despotovic present in [AD01] an approach to evaluate trustworthiness (i.e. the combination of credibility and performance) of peers based on the complaints posed for them by other peers following transactions. The approach also aims to provide incentives for truthful submission of complaints. The main idea is that a peer is considered less trustworthy the more complaints he receives or files. An agent trusts another if the latter is at least as trustworthy as the former.
- Feldman et al. address in [FLS+04] the problems of free-riding (i.e. poor performance) and misreporting of feedback on contributions (i.e. low credibility) by an indirect reciprocity scheme. Their objective is for each peer to offer to any other peer roughly equal benefit as indirectly offered by the latter to the former. However, their approach provides opportunity for peers to lie about the contribution of others.
- Ngan et al. have proposed in [NWD03] another indirect reciprocative approach for avoiding free-riding and false claims in a peer-to-peer system for sharing storage capacity. This approach requires peers to publish auditable records of their capacity and their locally and remotely stored files. However, collaborated adversaries can exploit this mechanism by claiming to have stored huge files of one to another.
- Finally, in [MG04], Marti and Garcia-Molina showed that it is preferable to rely on the referrals of friends for selecting providing peers (i.e., nodes for which there exist direct own trust values) rather than on those of the distant node's neighbors.

It is apparent that most of the above mechanisms apply to either specific peer-to-peer systems or to more general ones under specific assumptions. Moreover, these mechanisms aim to provide incentives at the application layer, *e.g.*, for contribution, for high application-level quality etc. Therefore, such mechanisms could be considered for SmoothIT only if they are put in the context of ETM and combined with "lower-layer" performance indices.

Chunk selection strategies

- Random

  The easiest method to choose a chunk to upload is to select a random chunk available at the uploader that is missing at the downloader. This strategy however is prone to the problem of chunk starvation. Chunk starvation means that only a very small number of copies of a chunk are available in the overlay, while the other chunks are well spread. In the extreme case even no copy is left of the starved chunk. Peers that want to finish the download are normally just missing the starved chunk, which leads to a high competition at the few peers that can upload this chunk. As a consequence, the download times for that chunk increase. Moreover, if peers finishing their download do not act as seeders afterwards, the chunk stays starved.

- Rarest First (RF), Least Shared First (LSF)

  To prevent chunk starvation, overlays like, *e.g.*, BitTorrent, employ a Least Shared First (LSF) chunk selection strategy, also called Rarest First. This strategy chooses an eligible chunk that is spread least in the network. This works especially well in networks where an overview on the chunk population of a given file can be provided easily, as is the case for the tracker-based BitTorrent. If no entity which has all information about the chunk distribution exists, peers have to estimate this distribution. The error made by this estimation may again lead to an uneven degree of distribution between the different chunks, in the worst case again leading to chunk starvation.

- CygPriM

  The CygPrim strategy (cyclic priority masking) [SHT05] tries to keep the chunk distribution balanced although only locally available information is used. It does not serve any chunk requests, but uploads only one defined chunk at a time. When this chunk is downloaded, the next available chunk from the file is offered, and so on. Thus, a peer tries to upload the whole sequence of chunks consecutively. When the last chunk has been uploaded, the cycle begins anew. Chunks that are not requested by any of the peers are skipped. The strategy is robust against selfish peers (leechers) and offers download times in the same order of magnitude in a best-case scenario with benevolent peers.

- Closest Deadline

  For the adaptation of chunk-based distribution overlays to streaming applications, not only the availability of chunks is important, but also their usefulness for the peers. A peer that already started viewing content while downloading might have passed the point in time where a chunk would have been played out. Therefore, this chunk is no longer of interest for that peer. On the other hand, chunks that are nearing their playout time but that have not yet been downloaded have a higher priority to be downloaded.

  As a result, the chunk selection includes the deadline of chunks in its strategy. An example named BiToS is presented in [VIF06], where chunks are sorted in two categories, a high priority set and a low priority set. The high priority set contains chunks that are close to their playout time, which have a higher probability to be downloaded. For stability, the larger low priority set again follows the Least Shared First strategy.

Resource access methods

Resource access methods for self-organized overlay networks encompass the following functions [Tu05]:

- permission for access to resources,
- prioritization of access to resources,
- scheduling of access to resources.

In self-organized systems resource access is performed in a decentralized way. When assessing performance of resource access mechanisms the following measures can be perceived as most appropriate due to the user-centric character of resource access [Tu05]:

- time needed to exchange certain resources,
- throughput-related to certain resources download.

Relevance for SmoothIT

The chosen VoD streaming application is chunk- and MSD-based. Therefore, the resource exchange mechanisms described here are of high relevance for the simulation and implementation of a SmoothIT concept.

4.2 Self-Organization of Resources

Apart from the explicit requests of resources from other peers, different SOMs are employed in search and distribution overlays to ensure that data does not get lost due to churn, and to facilitate an easier access to resources by placing them intelligently in the network. In the following, some of these mechanisms are discussed.

4.2.1 Data Replication Strategies

Goals of replication

Replication is a strategy in which multiple copies of some data are stored at multiple sites [BHG87]. This strategy is responsible for the maintenance of on-line copies of data and other resources. Data replication on several hosts helps to improve system performance, as the number of hosts that have to be probed before the query is resolved is minimized. By storing the data at more than one site, if a data site fails, a system can operate using replicated data, thus increasing availability and fault tolerance. At the same time, as the data is stored at multiple sites, the request can find the data close to the site, where the request originated, thus, increasing the performance of the system. But replication introduces also overheads due to creating, maintaining and updating the replicas. If the application has a read-only nature, replication can improve the performance. But, if the application needs to process update requests, benefits of replication can be neutralized to some extent by the overhead of maintaining consistency among multiple replicas.

Replication scenario and challenges in replicated environment

Different combinations of events and access scenarios of data are possible in a distributed replicated environment. Depending on requirements, three types of replication scenarios can be identified [GB06]:

- read-only queries,
- update transactions,
- managing mobile clients.

For read-only queries, data can be accessed by a query without worrying about the correctness of the data. Data may be generated at some site and can be read by other sites.

In case of update transactions special consideration in the design is required. To maintain the consistency of data, the order in which the transactions are executed must be maintained.

Sometimes mobile devices do not have enough space to store the data, while at other times they need to access real-time data from the office. In these cases, data is downloaded on demand from the local server.

Challenges in replicated environment

Challenges in replicated environment can be summarized as follows [GB06]:

- *Data consistency*: maintaining data integrity and consistency in a replicated environment is the main problem. High precision applications may require stricter consistency of the updates made by transactions.
- *Downtime during new replica creation*: if strict data consistency is to be maintained, performance is severely affected if a new replica is to be created as sites will not be able to fulfill requests due to consistency requirements.
- *Maintenance overhead*: if the files are replicated at more than one site, it occupies storage space and it has to be administered. Thus, there are overheads in storing multiple files.
- *Lower write performance*: Performance of write operations can be dramatically lower in applications requiring high updates in replicated environment, because the transaction may need to update multiple copies.

Types of replication protocols

For performance reasons, the system may implement [GB06]:

- *synchronous* replication: all the replicas are updated before the transaction commits, updates to all replicas are treated in the same way as any other data item.
- *asynchronous* replication: only a subset of the replicas is updated, other replicas are updated after the transaction commits.

Another important aspect on which the replication strategies can be classified is based on the concept of primary copy. It is represented as:

- Group: Any site having a replica of the data item can update it. This is also referred as *update anywhere.*
- Master: This approach delegates a primary copy of the replica. All other replicas are used for read-only queries. If any transaction wants to update a data item, it must do so in the master or primary copy.

Replication strategies in P2P systems

Some data replication strategies in P2P systems can be identified [GB06]:

- Based on size of files:
  - Full file replication: “full” files are replicated at multiple peers based upon which node downloads the file. This strategy is used in Gnutella. This strategy is simple to implement. However, replicating larger files at one single file can be cumbersome in space and time [BMS+02].
  - Block level replication: Divides each file into an ordered sequence of fixed size blocks. This is also advantageous if a single peer cannot store the whole file. Block level replication is used by eDonkey. A limitation of block level replication is that during file downloading it is required that enough peers are available to assemble and reconstruct the whole file.

  - Erasure Codes replication: Provides the capability that the original files can be constructed from less number of available blocks.
- Based on replica distribution [CS02]:
  - Uniform: The uniform replication strategy replicates everything equally.
  - Proportional: The number of replicas is proportional to their popularity. Thus, if a data item is popular it has more chances of finding the data close to the site where query was submitted.
  - Square-root: The number of replicas of a file is proportional to the square-root of query distribution.
- Based on replica creation strategy:
  - Owner replication: The object is replicated only at the requester node once the file is found; *e.g*., owner replication used in Gnutella.
  - Path replication: The file is replicated at all the nodes along the path through which the request is satisfied; *e.g*., path replication used in Freenet.
  - Random replication: Random replication algorithm creates the same number of replicas as of path replication. But, it distributes the replicas in a random order rather than following the topological order.

Replica placement

For the purposes of file availability, peer-to-peer systems should not ignore the availability characteristics of the underlying workstations and networks on which they are implemented [BMS+02]. In particular, systems should recognize that there is wide variability in the availability of hosts in the system. According to [SGG02] fewer than 20 percent of Gnutella's peer systems had network-level availability in excess of 95 percent, while over half of the remainder had availability under 20 percent. Given such a wide variability, the system should not place replicas blindly: more replicas are required when placing on hosts with low availability, and fewer on highly available hosts. Moreover, under many predictable circumstances, peer failures may be correlated. The results of research done by [BMS+02] show that the number of hosts running Gnutella is well correlated with time of day. As a consequence, placing replicas in out-of-phase time zones may be a sound replication strategy.

4.2.2 Redundancy Methods for Overlay Networks

The goal of redundancy provisioning in overlay networks is related to fault-tolerance assurance. The availability to all resources is ensured only when a certain resource is not stored in one node, but is in some way stored in many sites simultaneously. The main difference of so-called redundancy methods for overlay networks with simple replica systems lays in the fact that for the latter systems we have a number of copies of protected resource (i.e. a file) distributed across the system. On the other hand redundancy in this context means that each object is divided in some number of fragments (m), which are by using erasure coding stored as n (n>m) entities across the overlay network [RL05]. The idea of erasure codes lays in allowing for item reconstruction on the basis of any m collected fragments. The advantage of this method above simple replication is that the storage requirements are lower across nodes. It also ensures the higher level of availability over replica systems when the nodes have lower reliability parameters.

4.2.3 Caching Schemes

While the topic of dedicated cache entities is covered in Section 5, we present here caching methods where peers themselves also provide the functionalities of caches.

Video caching schemes

Many caching schemes have been proposed to handle video [BPL+02]. Some schemes have inherited the main characteristics of web caching schemes and have videos as single entities which are either cached completely or not at all. More recent works take into consideration the special characteristics of videos: their structure, the associated rate variability, their large volume and the need for a time-constrained delivery of content.

In [GLL07] an efficient cooperative caching scheme for video-on-demand service over P2P networks is proposed. In this architecture, published videos are encoded into multi-layered source bit stream and split into many of segments, which are distributed to overlay peers. The main design philosophy is that outbound bandwidth from multiple peers who have viewed and cached the same title before can be aggregated to serve a single video streaming request.

Unlike traditional web caching, most of the video caching schemes do not conform to the dynamic nature of caching according to which cache contents are dynamically updated in a demand driven fashion. The performance of a proxy is characterized by its ability to reduce the amount of data that cross the backbone network and also by the ability to provide streaming services with acceptable initial delay. The overall performance is sensitive to sudden changes in popularity. If the cache does not detect such changes quickly, there could be substantial mismatch between the content of the cache and the upcoming content requests, leading to a low hit ratio and an increased initial delay.

Overlay Caching Scheme (OCS)

In [TT02] the concept of an Overlay Caching Scheme (OCS) is introduced. It is intended as a light-weight, dynamic, and efficient collaboration technique for wide-area video caching. OCS runs on any overlay network, taking advantages of its multicast capability. The proposed solution dynamically establishes a virtual cache structure from caching nodes along the video delivery path between a client and the video server. A caching node is an overlay node that has sufficient storage space and is capable of serving video streams. OCS effectively utilizes virtual cache structures to deliver cached data to nearby clients, minimizing the service delays, server load, and network load altogether. The collaboration protocol offers better performance when compared to traditional video caching schemes. Furthermore, a simple analytical model is available to determine the best caching node in a virtual cache structure to store the cached copy of a video such that the average latency of requests served by the virtual cache structure is minimized.

In [GES05] semantic caches deployed on top of the super-peers architecture to improve the performance and scalability of the search in a P2P network are proposed. The system does not make any assumptions according to the type of the content shared in the network, and does not require users to explicitly specify their interests or manually categorize the content.

The semantic caching model for a content sharing P2P network is based on three main issues:

- the heterogeneity of the nodes is factored in by placing the semantic caches of file pointers only at the super-peers, which are selected nodes with higher capacities. These file caches are used by multiple peers at the same time, which shortens the cache warm-up period and increases the utilization of the cache.
- weak peers maintain caches of super-peers which are most likely to know the location of the files they are interested in.
- the mixed cache-management policy proposed achieves much higher cache hit ratios for less popular files while still being very efficient for the most popular items.

4.2.4 Relevance for SmoothIT

It remains to be seen how the availability of content is influenced by the architectural changes introduced by a SmoothIT solution. Therefore, it is not yet clear how important replication and redundancy will be in this context.

4.3 Monitoring of Overlays

Following the introduction, overlay size, churn rate, document availability, and a number of additional dimensions are outlined, which are of key relevance for monitoring an overlay.

4.3.1 Introduction

In this subsection, we consider mechanisms that monitor some aspects of the complete overlay in a distributed fashion. Based on the locally available measurement realizations, each peer can adapt parameters, *e.g*., the length of its neighbor list, to react on changes in the overlay. No manual interference is required; an automatic adaptation to changes in the system situation happens.

In contrast to a pre-defined setting of these parameters with no change during the lifetime of an overlay, this allows for a tuning of the overhead generated by SOMs such as content replication. If no such parameter adaptation happens, the parameter settings have either to be chosen for the envisioned worst case, generating more overhead during times when the situation is less severe, or for an average situation, running the risk of losing *e.g*., stability if the worst case happens. Therefore, a monitoring mechanism can enable or improve SOMs, since not only actions can be taken in a self-organizing way, but also the reactions can be observed in the same way.

Another alternative is a central entity measuring the overlay status, computing new global values for the parameters in question and distributing these results. However, the additional overhead introduced by having to collect the measurements from the peers, which are still necessary, a slower reaction time after changes and the higher failure risk of this method are disadvantages.

There are several parameters of a peer that depend on the actual network situation. One is the size of replication groups and the frequency with which documents are replicated. In networks with a larger churn rate, this frequency should be higher to prevent document losses. The same goes for stabilization intervals if specific regular routines are called to maintain stability. The size of a peer's neighbor table in some overlays depends on the size of the overlay to optimize routing and overlay paths.

The network characteristics that should and can be monitored to tune these parameters are described in the following.

4.3.2 Overlay Size

In some overlays, certain parameters depend on the overlay size, *e.g.*, in Chord, the number of direct successors a peer should know is log(n), where n is the number of peers in the network. Another example is the number of virtual nodes a Chord peer simulates.

In [BSH05], an algorithm is described that utilizes information locally available at one peer to estimate the size of the network.

More specifically, the distances between the successors of a peer in the id-space and the distances of the local peer's ideal finger positions to the actual peers they point to is used to estimate the density of the peer population on the Chord ring. The confidence levels of the used maximum-likelihood estimator (MLE) can be adapted to tune the algorithm for different reliability needs.

An overlay-independent method to estimate the overlay size is described in [MMK+06]. It is an active method, i.e., it uses probes to collect samples from the network. Two approaches are described, one using a Random Tour (RT), the other a Sample&Collide method. Each peer has to conduct his own measurements, which incurs potentially large overhead.

4.3.3 Churn Rate

Churn, i.e., the process of peers joining and leaving the network, is one of the biggest threats to the stability and efficiency of overlay networks. Many overlays have built-in mechanisms to counter these negative effects, *e.g.*, replication groups or stabilization algorithms. These mechanisms normally can be tuned to the level of churn the overlay encounters, if the current churn rate is known.

To monitor this, an algorithm was proposed in [BL07] that uses local observations at a peer in order to estimate the overall churn rate and to set local parameters like the replication group size or the synchronization intervals of a stabilization algorithm. The method uses the overlay as memory for the estimates and is passive, i.e., no extra measurements have to be taken in order to generate a result.

The basic principle is that each peer observes one defined neighbor in the overlay and records the total time it was online. Since it can not be guaranteed that the observing peer joined the network before the observed peer, the latter informs the former about the time it joined the network.

Additionally, the offline time durations of the peers are reported by joining peers to all neighbors it contacts while joining the network. From the online and offline interval instances, the churn rate can be estimated. The recorded observations are stored in a history at each peer and sent to new peers in the network to allow them to set their parameters. The size of the history provides a trade-off between accuracy (longer history) and responsiveness (shorter history).

4.3.4 Document Availability

One way to monitor the number of copies of a given document is to track the peers that store this document. When one peer in this set leaves the system, another copy of the document can be made in order to sustain the desired availability. This is especially viable in structured overlays, where nodes responsible for a document can be determined by their ID. An example is Kademlia, where the 20 nodes closest to a document in the ID space are that document's replication group.

When a peer in this replication group has not received the document for a fixed time interval $T_{republish}$, it distributes the document to the replication group, which may now contain different peers if churn occurred. A peer receiving the document resets its timer for the republish interval. An improvement lowering the overhead of this method was proposed in [BS07].

4.3.5 Query Rate, System Capacity, and Number of Shared Items

We first briefly outline motivation for why one would want to estimate these quantities. Consider a superpeer based overlay in which we want to tune the total number of superpeers in order to minimize the lookup traffic. First, one can observe that the traffic is an increasing function of the fraction of superpeers. So minimizing the traffic implies pushing the superpeers to their load limits in the sense that only a selected number of them should operate as superpeers and the rest should be demoted to leaves. A possible way to do this is computing the dependency of the load of superpeers on their fraction in the network and tuning the number of superpeers as to set the individual load as close to 100% as possible [ZDK06], [ZDK08]. To establish this dependency, one needs to know the parameters such as query rate, mean peer capacity and total number of shared items.

A simple method to assess the query rate in the entire network can be described as follows. Each peer *i* can monitor the total number of queries it has to answer within a period of time and thus estimate its own query rate R_i. It can send this value to a number of its neighbors and also receive from its neighbors their corresponding observations. These values need not be sent separately, they can be piggybacked with other messages the peers anyway exchange (*e.g.*, those of the stabilization protocol) in order to save bandwidth. Once the values of the neighbors are collected, peer *i* can compute the average $R_{i,avg}$. Having an estimate of the total network size N', the peer can compute the system query rate as $R = N'R_{i,avg}$.

Similar procedures can be applied in order to compute the mean peer capacity (actually, without multiplying with N') and the number of shared items.

The procedure we just described assumes no knowledge on the distributions of the involved parameters. Any available knowledge can be used to improve the estimation quality. For example, if we know the mean value of the number of shared items per peer then there is no need to use the sample from neighbors. That value can be multiplied with the estimate of the system size to give the total number of shared items.

4.3.6 Relevance for SmoothIT

The presented methods are mainly applicable for DHTs, which currently do not play a central role in an envisioned SmoothIT solution [SI08]. However, it might be necessary to gather information from the overlay considered in the solution. In this case, the more general techniques described are of use.

5 Other Mechanisms

In this section, we will list mechanisms and architecture additions that are not strictly self-organizing, but that are nevertheless also able to contribute towards an overlay implementing ETM.

5.1 Dedicated Caches

In computer science, a cache is a collection of data duplicating original values stored elsewhere or computed earlier, where the original data is expensive to fetch (owing to longer access time) or to compute, compared to the cost of reading the cache. Once the data is stored in the cache, future use can be made by accessing the cached copy rather than re-fetching or re-computing the original data, so that the average access time is shorter.

By deploying caches in a network, network administrators can manage the location of information that is frequently requested by clients. Among the well known impacts of caching on network and system performance are: a reduction of network traffic, a reduction of average client delay, and a reduction of origin server load.

To gain a maximum benefit from this, traffic should be attracted to these caches that would otherwise be costly, i.e., traffic otherwise routed to distant networks, which is both costly for an ISP as well as for the end-user (in terms of latency). To efficiently reduce the load on the original sources, caches ideally store the most frequently requested files, i.e., the most popular files. Full copies of these files should be stored in order to prevent incomplete downloads due to chunk starvation in case of MSD (Multi-Source Download).

For the same reason, caches are placed at major aggregation hubs in a network to be able to serve many requests. Also, placement at or shortly before peering points enables the cache to deflect traffic that otherwise would have been routed to another provider.

If caches are not part of the original overlay, *e.g.*, placed by a provider, they have to inspect messages deeply to be able to answer requests instead of the peer that was originally the destination. Also, it should have access to statistics about file requests in order to judge which files it should store.

To maintain copies of documents across the Internet several strategies can be used [PKS00]. *Caching strategies* create a copy when a user first requests the document; subsequent requests can be resolved locally. Cached copies can be destroyed at any moment, for example to reclaim storage space. *Replication strategies* create copies *a priori* as opposed to cached copies. Such replicas are intended to persist for a long time. When a document is modified, its copies must either be updated or destroyed so that users do not access stale data. Most caching policies apply diverse heuristics to guess when the original document has been updated. When a cache suspects that a copy is stale, it can destroy it or check for its validity. Replicas use a different strategy: they assume that the main server sends a notification when an update occurs.

Similar to web caching mechanisms (see below), dedicated caches in content distribution overlays are used to place content close to the potential consumers. Additionally, caches normally offer a better connectivity, increasing the end users' QoS and strengthening a distribution overlay.

In [OAM+05], among other additions to an eDonkey-based network a cache is described that supports content distribution when mobiles in a radio provider network are part of a content distribution overlay.

Web caching

Web caching is the caching of web documents (*e.g*., HTML [Hypertext Markup Language] pages, images) in order to reduce bandwidth usage, server load, and perceived delay. A web cache stores copies of documents passing through it; subsequent requests may be satisfied from the cache if certain conditions are met.

Web caches can be deployed in different ways:

- User agent caches, such as those in web browsers, are *private caches*, operating on behalf of a single user. Intermediaries can also implement *shared caches* that serve more than one person.
- Proxy caches, also known as *forward proxy caches*, are usually deployed by Internet Service Providers, schools and corporations to save bandwidth. Transparent proxy caches are a variant that does not require clients to be explicitly configured to use them.
- Gateway caches, sometimes known as reverse proxy caches, *surrogate caches*, or web accelerators, operate on behalf of the origin server. A number of gateway caches can work together to implement a Content Delivery Network.

In designs for web proxy caching, documents replication and distribution is an important element. Documents are placed in different proxy servers in order to reduce network latency. Two caching architectures have been proposed to enable proxy caches to collaborate [W99]:

- hierarchical caching: caches are placed at multiple levels of the network. When caches of one level cannot satisfy a request, the request is directed to the caches at an upper level.
- distributed caching: there are only two levels of caches - browser caches and proxy caches. When a request encounters a browser cache miss, it goes to a proxy cache, and proxy caches serve one another’s misses. In order to decide which proxy cache to go to retrieve a missed document, all proxy caches keep meta-information about the content of the other cooperating servers.

Cache replacement algorithms

Cache replacement algorithms [BPL+02] utilize the request history to estimate the current request probabilities and self-organize accordingly. Given a stationary request pattern and a large number of request samples, the underlying request probabilities can be "learned" by counting the requests for each asset. Replacement algorithms are able to provide a good estimate of the request ranking without the need to count a large number of requests, *e.g*., LRU (Least Recently Used) simply replaces the least recently used object upon a request of a new object.

In web caching the arrival of a single request changes slightly both the recent request history and the state of the cache, since the size of an ordinary web page is very small compared to the capacity of the cache. In video caching a single replacement causes a relatively greater change to the state of the cache, although a single request has similar impact on the recent request history as in web caching. Chunk-based replacement strategies try to establish a „Web-like" relation between a single request and the corresponding replacement unit.

5.2 Proxies

Instead of end systems acting as peers in an overlay network, they can use proxies to represent them. Depending on the type of end systems considered, this may offer several advantages. One is the fact that a number of peers may be served by one proxy, lowering the number of peers and therefore the maintenance effort of the overlay.

Another advantage would be the potentially higher resource availability at a dedicated proxy in comparison to end hosts with low resources, *e.g.*, mobile peers. Also, a proxy may be more reliable than an end system, enhancing the overlay stability.

Employing proxies of course incurs higher costs for their provider. However, he is able to influence the placement of the proxies and possibly the association of peers with a specific proxy to influence the traffic streams between an end user and a proxy on one hand and between proxies/peers on the other hand.

Additionally, a combination of the functionalities of a proxy and a cache is possible, combining the advantages of both.

5.3 Crawlers

Crawlers or crawling peers were introduced in [HMT05] as supplements to overlays containing mobile peers, i.e., peers with few resources. A crawling peer searches for files on behalf of these peers and thus reduces the signaling traffic that is costly to the user (and in case of mobiles also to the radio network provider). Additionally, the influence of the transient peer state of the mobiles is diminished. An added benefit is the information gathered at the crawling peer about the popularity of files, which can be used by the caching scheme described above.

5.4 Direct Insertion of ISP-owned Peers in the Overlay

The introduction of ISP-owned peers in the overlay could imply performance improvements both for the ISP and the end-users. In order to achieve these improvements the ISP-owned peers' insertion can be combined with mechanisms that exploit locality. There are two possible scenarios:

- Enforce locality

The main idea is that the ISP designates a peer of its own as a so-called "gateway" peer. All peers inside the ISP connect to each other and to the gateway peer, but they are enforced not to connect directly to other peers outside their ISP. Only the gateway peer is able to connect to peers that belong to other ISPs. In order to avoid the direct connection of its peers to external peers, an ISP must intercept its peers' messages directed to external peers or trackers and manipulate them before sending them to the external world. For instance, in case of BitTorrent, peer p that belongs to ISP A starts its client using as input a torrent file. The tracker t that serves the content file that this torrent describes belongs to a different ISP B. When p sends a query to tracker t, ISP A must be able to intercept p's query. This is carried out through special devices that use deep packet inspection to identify P2P traffic and are situated along the edges of the ISP's network. After intercepting p's query, the query is sent by the intercepting device to tracker t as a query originated by the gateway peer. Of course, when tracker t sends back its reply, the gateway peer will receive it and will join the swarm in order to download the requested file. During downloading and after it is over the file can be uploaded by the gateway peer to any internal peer that has asked for this file including p. The drawback here is that such solutions are not always possible to deploy and maintain.

- Combine with locality awareness

The introduction of ISP-owned peers can also be combined with locality awareness, which is considered a promising solution for the reduction of inter-ISP P2P traffic. By the term locality awareness is meant that each peer chooses as much as possible its peers from those within the same ISP instead of choosing peers belonging to different ISPs. Such a mechanism can be implemented by applying minor changes to the client and/or the tracker. In the first case, the client must be able to identify from the tracker's reply those peers that belong to the same ISP with it and choose them. To achieve this, the client must be provided with certain information. The challenge here is also to provide incentives to the client to choose the specific peers even if the download rates it experiences from them is lower compared to others. In the second case, the tracker replies to the client's query with a list of peers most or all of which belong to the same ISP with the client. The challenge with this approach is to inform the tracker as well about locality. Here, the ISP should inform the tracker about its own peers. As a result, any time that the tracker receives a query from a peer of that ISP, the tracker's reply will include the addresses of the specific ISP-owned peers. In both cases, the introduction by the ISP of a peer that will be appearing frequently in the replies of the trackers will influence the generation of traffic, which will be more biased towards locality.

Especially in the BitTorrent case, even if we do not assume locality awareness, the introduction of ISP-owned peers with high upload/download bandwidth in combination with the tit-for-tat incentive mechanism is expected to have as a result performance improvements both for the ISP and the end-user. The insertion of an ISP-owned peer means that each requested file is downloaded only once from networks outside the ISP. After the ISP-owned peer has downloaded a copy of the content file, it can start uploading it to local peers. As a result no redundant ingress-ISP traffic is generated.

Obviously the ISP-owned peer cannot be a regular peer, otherwise, if it has the same upload bandwidth as other peers, then the downloading time would be increased significantly. The ISP-owned peer needs to have much higher bandwidth than other peers, and as expected the required bandwidth grows as the number of the peers inside the ISP increases.

However, in the BitTorrent case, an important issue arises. As was mentioned before, it is necessary that the ISP-owned peer has high upload bandwidth so that the download rates of the internal peers do not decrease significantly. Because of both the tit-for-tat mechanisms and its high upload bandwidth, the ISP-owned peer is likely to be also selected by external peers to download from. As a result, egress-ISP traffic is generated. Furthermore it is observed that if only an ISP A uses the gateway peer approach with a high bandwidth peer, while all other ISPs do not, then peers outside ISP A could also finish downloading faster than peers within ISP A. The reason for this behaviour is that the

ISP-owned peer has little to no content to gain from the internal peers, and via the tit-for-tat mechanism, would rather exchange blocks with external peers. Namely, the extra resources used by ISP A benefits peers that belong to other ISPs, certainly an undesirable result for ISP A.

In addition, there are important scalability issues. Using a single ISP-owned peer that connects to external peers can result in a significant increase of download time for the internal peers. This is the case even if the designated ISP-owned peer has high upload and download bandwidth. The problem gets heavier as the number of the internal peers increases. A solution to that is the insertion of multiple ISP-owned peers. Instead of having only one, an ISP can use more ISP-owned peers that are connected to the outside world and have high upload bandwidth. However the number of ISP-owned peers cannot increase unreasonably. The introduction of multiple ISP-owned peers improves performance yet at an increase of the cost. The tradeoff here lies between the decrease of intra-ISP traffic and resources consumption within the ISP and the increase of the inter-ISP traffic as it was discussed before.

The important questions here are: How many ISP-owned peers achieve optimal performance for both the ISP and the peers? What should be the relation of the number of ISP-owned peers and the ISP's size, *e.g.*, number of regular peers? How much upload bandwidth should be available to the ISP-owned peers?

Also in this solution it is important to take into account legal issues, the ISP owned peer cannot download/provide illegal contents.

5.5 Community Structures

The community structure mechanism is a supporting mechanism for handling path changes and self-organization of resources. In both cases the aim is managing the overlay networks in a self-organizing way. A physical network is divided into groups of nodes which are called communities. In this way a two level hierarchy for managing networks is obtained. The management is supposed to be performed in a hierarchical way. As an example consider the following situations in a network. One or a few nodes experience congestion problems, these problems appeared in the presence of peer-to-peer traffic. This traffic should not be discarded, the troublesome traffic should simply be routed through other regions of the network. A particular problem is localized in some clusters (communities). In the first place there will be attempts to handle traffic inside a particular cluster which has experienced traffic problems, in the second place actions will be undertaken on cluster level.

Traffic can be moved to a particular region of the considered network by changing the path or by attracting the traffic to this region. The first method described in the following relates to the path handling mechanism, the second one relates to data replication strategies (section 4.2). In the considered replication mechanism some peers are added to the overlay network, they are equipped with peer-to-peer applications and some data with artificially increased popularity parameters.

These peers are located in a particular community and they are attraction points for particular traffic (for a specific peer-to-peer application). In this scenario, communities are built in such a way that the specific traffic is confined to a particular cluster or group of clusters. The algorithm for establishing communities has to be proposed. In contrast to the path handling mechanism, in this case the overlay network is directly influenced.

For both mechanisms, clusters are introduced in order to limit the number of interacting nodes and localize some tasks or traffic in a part of a network. Clusters are created according to some algorithms which take into account specific creation strategies.

5.6 Mechanisms for Handling Path Changes

In this section a mechanism is described for managing an overlay network by making changes in the underlay network. This path handling mechanism does not interfere directly with the overlay network. In this mechanism a path in an operator domain is rerouted, thus the operation takes place on the level of an underlay network. Peer-to-peer traffic is considered in the presence of other type of traffic.

By changing the path in a physical network overlay networks are influenced, in the sense that some peers, after the change may not be available or are not popular for partners in a peer-to-peer network.

The strategy is delivering as much traffic as it is possible in the actual state of an operator network. Conditions in a network changes dynamically and the mechanism reacts to these changes as fast as the mechanisms employed by the operator allow. If some part of an operator network does not provide a sufficient amount of resources, a part of or the whole traffic is moved to other regions of the operator’s network where these resources are available. If resources are not available or if there is no alternative path in the network, this traffic is discarded.

The path change mechanism can be also used in the case when specific traffic (for instance voice) increases and there are not enough resources on the actual path for this traffic. Suppose that there are resources on this path that are allocated for another type of traffic. Some resources can be freed by changing the path of that other traffic and reallocate these resources to voice traffic.

A component of the path handling mechanism is community structures (described in the previous section). Taking into account actual conditions in a provider network, the suggested mechanism is responsible for the structure of communities. After creating communities attempts are made to handle paths.

As a first step, community structures (also called clusters) are found and established in a network. Nodes are grouped (routers, hosts) in order to make traffic engineering on the level of this group. The traffic engineering decision should be undertaken in a self-organizing way. By introducing clusters the traffic engineering is organized in a hierarchical way. If some traffic problems are discovered in a particular region of the network, these problems are localized in a particular cluster. It is expected that the cluster makes attempts to solve these problems in its own domain. This can be achieved by moving some traffic to another path inside the cluster, requiring that these paths must belong to this cluster. If the cluster has exhausted all its resources, it should pass the traffic to one of the neighboring clusters. If this is impossible, it should communicate with a cluster which is a source of the traffic which causes problems and the source cluster has to take over the procedure for handling the troublesome traffic in a similar way.

In this moment the decision is moved on the inter-cluster level. The source cluster tries to send some part of the traffic to other neighboring clusters.

Different aims can be followed in organizing the network clusters. For instance in the case of peer-to-peer networks, we would like to keep this type of traffic in a particular part of network thus forming clusters of nodes and giving special rules for border routers (the cluster border) which does not pass the traffic for some other parts of network. In order to keep traffic in a cluster we can introduce some points of a traffic attraction which can be powerful servers. These servers should store files or parts of files which are very often exchanged.

In the rest of the description an example of the network with a cluster structure is given and some ideas how some traffic engineering decisions can be undertaken.

The criteria for discovering the cluster structure are not discussed; we assume that we have established such a structure by some methods.

In Figure 3 a network model is shown in which points represent routers. Green points refer to core routers, blue points represent border routers and red points represent routers which connect subnets hosting end clients.

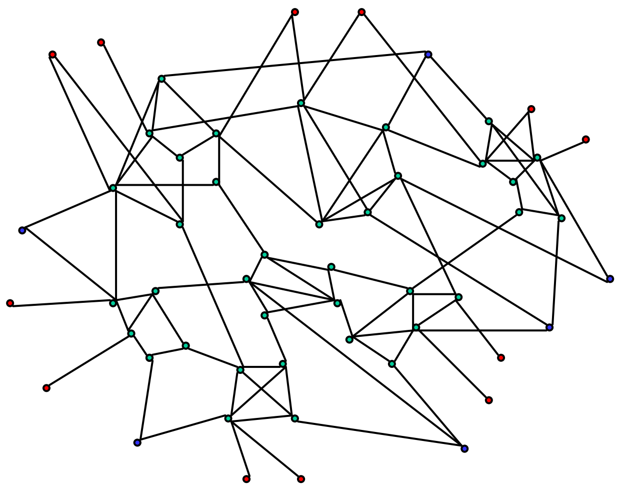


Figure 3. The network model

As it has been already mentioned a cluster structure is built in this network (see Figure 4).

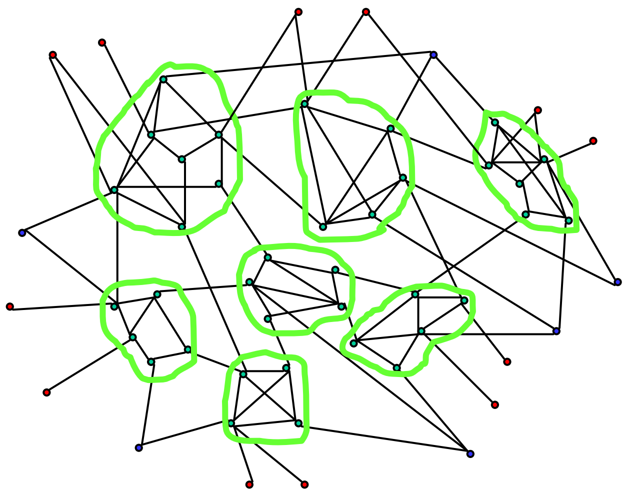


Figure 4. The cluster structure in the network

In Figure 5 a traffic path is marked from one domain to another. The packets try to follow the shortest path (the red path). One router on the path has problems with delivering packets (the arrow indicates this router). In a first step, this cluster tries to solve the problem by what is represented as a blue path. Because the cluster could not manage with the problems it makes an attempt to send traffic to the neighboring cluster which is able to solve the problem (the violet path).

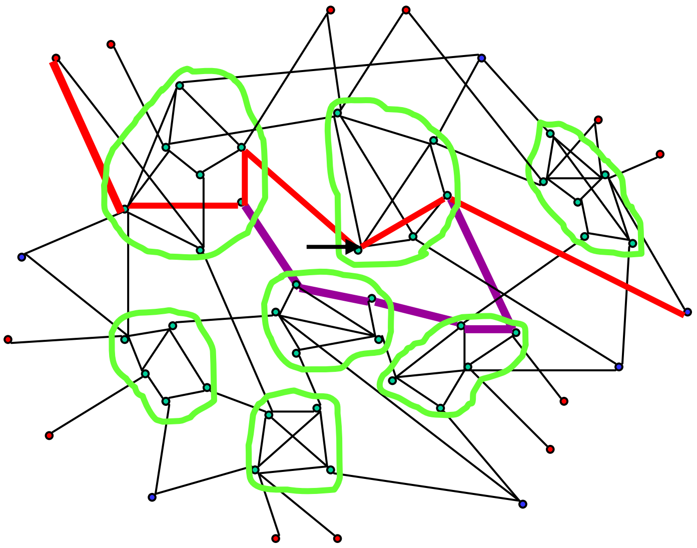


Figure 5. Path changes in the network

A potential mechanism for handling the path change may be implemented as follows. This mechanism operates independently from routing. Standard routing protocols work in the network. Consider a router with one interface for input traffic, and two interfaces for output traffic (see Figure 6) Routers can have arbitrary numbers of input and output interfaces, this restriction is made for simplicity.

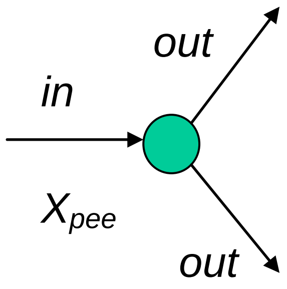


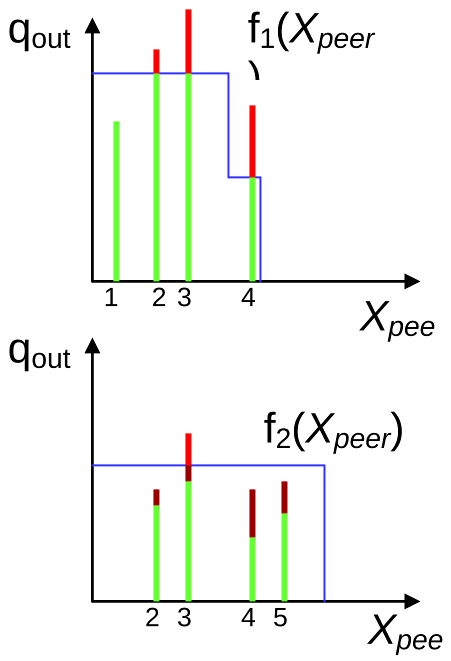
Figure 6. Router interfaces controlled by the action function

By *in1* the whole input traffic on this interface is denoted, X_{peer} is a part of input traffic *in1*. X_{peer} can be peer-to-peer traffic. *out1* and *out2* refer to the output traffic on two output interfaces respectively.

Suppose that the whole X_{peer} traffic is passed to out1 interface (for routing reasons). We want to have influence on the X_{peer} traffic in the sense that we can move some part of the X_{peer} traffic to the other interface – out2.

For these purposes potentials are defined, or reaction functions. The shape of this functions dictates when some part of the interesting traffic is passed, in our case X_{peer} traffic, to the interface out2.

The reaction functions are presented on the charts below.



q_{out1} – represents a load level of the output queue on interface1 as a function of the X_{peer} traffic. This load represents total load on that interface (X_{peer} + rest).

$f_1(X_{peer})$ – represents the reaction function for the interface out1 (the blue curve). In the same way, we describe the chart for interface out2. The meaning of reaction functions will become clear in the following description.

In the presence of particular X_{peer} traffic we have the specific load of output queues, which is denoted on the charts by vertical bars. The main output interface is out1. By numbers on the X_{peer} axis some specific points which relate to particular traffic are enumerated.

For traffic 1 we observe that queue load for the interface out1 is below f_1, it means that we do not have any problems with X_{peer} traffic. For point 2 we experience problems, the total queue load exceeds the value of f_1, so the X_{peer} traffic has to be moved to interface out2. By moving traffic we have increased traffic load on the interface out2, but it does not exceed the allowed level which is represented by the value of the reaction function f_2. In the case of point 3 some part of traffic has to be discarded. In the case of point 5, the shape of the reaction function f_1 forces to move the X_{peer} traffic to interface out2, and the f_2 allows to make this procedure. We can see on the chart that there is such a value of X_{peer} traffic that this type of traffic has to disappear from both interfaces. For instance we can expect that such level of this type of traffic can generate exponential increase of responds so the traffic is stopped.

Self organizing procedures may be found which allow adjusting the shapes of the reaction functions in an optimal way. Optimizing procedures can be applied on the cluster and inter-cluster level. The aim of the optimization can be different, one time we want to maximize background traffic in the presence of X_{peer} traffic, in other situation we want to maximize X_{peer} traffic. Optimization procedures should relate also to a cluster structure. The conditions in the network change dynamically, so we expect clustering procedures and traffic engineering procedures to be a dynamic operation.

5.7 Pricing in Support of ETM

A pricing scheme defines the charge calculation rules and prices to be used for the charging for service usage, i.e. it specifies how to calculate the charge for a given service usage. There are several different pricing schemes proposed in the literature [FDL00], considering different charging methods and price discrimination for IP-based services. A pricing scheme should ideally meet different requirements, including economic efficiency (i.e. recover costs and maximize revenue), network efficiency (i.e. maximize network utilization and avoid congestion), support of QoS provisioning, transparency, predictability, fairness (i.e. divide costs fair among users), social fairness (i.e. do not exclude users), and finally technical feasibility.

Pricing can be used as an incentive mechanism for users in order to influence user behavior. Thus, it can be applied *e.g.*, for congestion control or QoS provisioning. It can also provide the right incentives for users to participate in an ETM mechanism.

Flat Rate Pricing

In case of flat rate pricing a fixed amount of money per time unit (*e.g.*, monthly) is charged independently of the actual service consumption. Flat rate pricing has several advantages. It is simple and convenient for both users and operators, since users can easily predict the final charge to be paid and operators do not need traffic measurement for charging and billing. Furthermore, it is considered socially fair, because any user can access the network while receiving the same service level. However, flat rate pricing does not provide economic incentives and feedback to the user in order to adapt his behavior and therefore it cannot be used for congestion control or service differentiation (*e.g.*, QoS enabled services), since users would always use the best available service or QoS class.

Time and Volume-based Pricing

Opposite to flat rate pricing usage-based pricing schemes consider individual service and resource utilization. The time-based and volume-based pricing schemes are two widely applied usage-based pricing schemes. In case of the time-based pricing scheme charging is performed according to the duration of the service usage and therefore it is especially suitable for circuit switched networks, where resources are assigned to a user, *e.g.*, in case of a traditional telephony service. It is easy to implement and users can predict their charge.

In case of the volume-based pricing scheme the user is charged for the data volume being transferred and therefore it is better suited for packet switched network technologies. The volume-based pricing scheme is easy to implement and charges are predictable for the user. Additionally, it provides incentives to users to adapt their behavior and to use network resources more efficient. Both the time-based and volume-based charging scheme can support service differentiation if different prices are set for different services.

The Cumulus Pricing Scheme

In case of the cumulus pricing scheme [RHS03], a contract is negotiated between the user and the operator that specifies the user’s requirements in terms of bandwidth and his flat rate price. It is allowed that the actual bandwidth usage of the user varies over time, but in that case the user receives a feedback from the network in the form of cumulus points. If the bandwidth usage exceeds the predefined value in the contract with a certain limit, the user gets red (i.e. negative) cumulus points. If the user does not use all the bandwidth, he gets green (i.e. positive) cumulus points. Positive and negative cumulus points are summed up and if the result exceeds a certain threshold over consecutive charging periods, extra fees are charged for overusing the network and the contract will be renegotiated. The cumulus pricing scheme is technically feasible and provides a trade-off between economic efficiency, simplicity of the accounting technology involved, and transparency and predictability for the user.

Priority Pricing

Priority pricing [CES+91] enables the assignment of different prices for traffic with different priorities. In a network that supports data transmission with different priorities, each packet indicates its priority in its header field. Packets are charged according to their priority, where the higher the priority the higher the price for the user. According to [CES+91], users can choose priority classes in a short time frame, enabling traffic management and congestion control. Since prices of priority classes reflect the users' utility, this pricing scheme increases overall economic efficiency.

Comparison

Comparing and analyzing the above pricing schemes with respect to ETM support, it is important whether a pricing scheme can be used to provide incentives and to give feedback to users in order to change their usage behavior. Since network operators can reduce their costs by keeping network traffic in their own network and decreasing the amount of inter-domain traffic, operators would like to give their users an incentive that they generate rather intra-domain than inter-domain traffic. One possible solution is to provide this incentive through pricing. Therefore, the comparison in *Table* 2 presents whether a pricing scheme is suitable to support locality-based pricing.

Table 2. Comparison of pricing schemes

Pricing Scheme	**Support of ETM**	**Accounting requirements**
Flat rate	Not suitable for ETM, since cannot be used for service differentiation (*e.g.*, locality-based, QoS-based)	No accounting required
Time-based	Not suitable for packet-switched services, therefore not relevant for ETM	Accounting for time per user
Volume-based	It is suitable for ETM if different prices are applied, *e.g.*, for intra and inter-domain traffic or for different QoS classes	Accounting for traffic volume per user and service type
Cumulus pricing scheme	It is suitable for ETM if cumulus points are given for each service type (*e.g.*, intra or inter-domain traffic) separately	Accounting for traffic volume and manage cumulus points per user and service type
Priority pricing	It is suitable for ETM if priority represents the locality information	Accounting per priority and per user

A pricing scheme has to be able to differentiate between intra and inter-domain traffic (in general the destination of traffic) in order to change user behavior and support ETM. Therefore, a new approach is proposed – called locality-based pricing – that requires existing pricing schemes to be extended and/or applied with respect to locality information. According to *Table* 2, volume-based pricing, the cumulus pricing scheme, and priority pricing can support the locality-based approach if they consider the destination of traffic as a new parameter in the price definition.

5.8 Relevance for SmoothIT

The mechanisms listed in this section require a heavy intervention of an ISP in the overlay network. Some of the presented solutions touch the topic of network neutrality, since traffic has to be inspected deeply, *e.g.*, in the case of caches. Also, the techniques described in 5.1-5.4 may introduce legal problems due to the storage of content on ISP-owned machines without the provider having full control over what content is stored there. While pricing and path changes might be used as incentive mechanisms, the rest may be an additional support for the SmoothIT architecture. However, it does not seem to be in the focus of the project work.

6 Interworking of Self-Organization and Economic Traffic Management

The purpose of the interworking between ETM and SOMs is the improvement of all players' payoff. That is, the end-users, the overlay provider and the ISP should all benefit from such an interworking, which should thus lead to a win-win-win situation.

Realization of ETM can be divided into two major categories of approaches based on the "*transparency*" of those mechanisms as it is seen from the point of view of the overlay. In particular, the first category comprises non- transparent ETM approaches where ETM is not enforced, but there are incentives given to the overlay provider in order to adopt ETM with them, *e.g*., by means of price differentiation, QoE, etc. The second category comprises transparent (for the overlay provider ETM) approaches where the overlay provider is not involved; on the contrary either the user is given incentives to alter his behavior according to the information provider by ETM, or ETM is performed "directly" by introduction of hardware components or other network entities, etc.

Three major approaches that reflect the above discrimination are described below. Each of them is illustrated by means of an example. To this end, we choose the application class of overlay content distribution, with the application BitTorrent providing the basic architecture. Indeed, since content distribution generates a large amount of traffic and since traffic that has to be routed to foreign ISPs creating costs for an ISP, one of the possible ETM aims in this case would be to keep as much data traffic in the local Autonomous System (AS) as possible. In fact, besides the overall penetration of BitTorrent and attention of researchers it has attracted, it is also the basis of the applications selected for the trials (cf. D1.1, Section 7.3).

1. Interworking of SOM and ETM performed by the overlay provider – Non-transparent ETM

In this case, the overlay provider can be given incentives in order to modify his overlay protocol, although these incentives are rather indirect as it was mentioned in D1.1-Section 5. Here, the interaction between the overlay provider and the operator leads to a win-win situation. Indirectly, 'win' for the overlay provider implies also 'win' for the end-users. Actually, as remarked in D1.1-Subsection 7.2, there is rarely any conflict of incentives between the overlay provider and the user. In order to perform SOMs, it is necessary that the overlay provider is aware of underlay information, *e.g*., proximity measurements, RTT, link congestion, link costs, etc. This information is provided to the overlay provider by the ETM mechanism. The changes to the overlay protocol that are required have to be implemented, perhaps by means of plugins to the existing software, but have to be carefully done in order to be compatible with older versions of the protocol. In fact, provision of two versions of the software may lead to an even higher benefit for the overlay provider besides the extra satisfaction of the users. For example, he may introduce some charge to the improved version, while keeping the standard version for free. Another possible gain for the overlay provider can be imagined in a scenario where the content is originally offered by the overlay provider, *e.g*., software patches distributed via BitTorrent. In this case, a fast and efficient download directly influences the popularity of the content provider (which is here the overlay/tracker provider) with his customers.

In this context, we consider the BitTorrent example, with the objective to achieve as low as possible traffic in the inter-ISP link. To achieve this on the overlay level, peers should prefer download connections to other peers in the same AS instead of exchanging data with remote peers. The SOM used to achieve this goal is the neighbor selection done at the tracker. As already described, a new peer, who wants to download a specific piece of content, must contact the tracker for that file. The tracker assembles a list of peers and returns this list to the querying peer. The decision on which peers to include in that list can be made so that the ETM aim is supported.

To this end, information describing the underlay situation must be made available to the overlay. In this case, this is less complicated than in completely distributed scenario 2 (see below), since only the tracker has to be informed. A possible implementation of this information exchange could follow the pull model, where the tracker contacts a separate, ISP-provided information service with the IP of the peer requesting neighbor data. An example of such a service is presented in D1.1-Section 6 and can be employed both under this approach as under the next one, which is transparent for the overlay provider. The information service is also informed about all peers currently participating, and in case of peers connected via the local ISP also their location and other characteristics. It selects the peers it deems beneficial to both the network and the peer in question using a metric. Then it returns this list to the tracker, which may forward or modify it. To this end, a mapping of peers to ISPs has to take place, in order to allow the tracker to contact the information service of the correct ISP. Alternatively, one central information service might be created, with different implications for the distribution of provider-dependent information.

A metric reflecting the ETM purpose described could be the cost generated by a connection between the local peer and the peer that is evaluated by the metric. This need not be the actual money that has to be spent by the ISP to maintain that connection, but may be normalized. It also could reflect link utilization, i.e., less congested links are treated as less costly. In general, this metric should return better values for peers close to the local peer, i.e., that have short physical links in the same ISP network. The metric allows for an ordering of the peers the tracker knows with respect to the peer requesting a neighbor list. The *n* best peers are put on the list, perhaps along with a small selection of random peers in order to enhance the stability of the overlay.

As a result in the above example, the response times of the chosen peers should be shorter, and the available bandwidth in a link between the local peer and these peers could be higher, since physical links with low utilization are preferred. Data transfer connections between the local peer and his neighbors therefore may expect higher throughput. The incentive for the ISP to provide the cost information is the fact that if the overlay prefers to establish low-cost connections, the cost for the ISP to handle the traffic now flowing over these connections is lowered. On the other hand, the end user should be able to observe shorter download times, i.e., a better service quality. Since both parties involved gain an advantage by using the described system instead of the original implementation, it is likely to be accepted.

However, this example also raises specific questions, due to the additional party introduced by the tracker and the fact that the tracker has to be modified. As already mentioned, it is assumed that the overlay provider recognizes that it is beneficial for him to modify the tracker software. Yet, the tracker may be hosted by different entities. A change like the one described above necessitates an update in the tracker software, both in its "intelligence" and in its interfacing with the entities providing the necessary information. It has to be made apparent that a newer tracker version may indeed be more popular with users, so that the entity hosting the tracker switches to that version, either on his own or by getting this from the application provider.

2. Interworking of SOM and ETM performed by the end-users – Transparent ETM

In this case, we assume that the overlay provider is reluctant to modify the application protocol, although this does not necessarily imply that such a modification would not be beneficial to him. So, the interworking between ETM and SOM leads to a win-win situation for the end-users and the operator. Here, the end-users can be given incentives, *e.g*., QoE, price differentiation, in order to make different choices based on new criteria that would complement or substitute the existing ones. Again, all the information that is necessary is provided to the end-users' clients by the ETM, *e.g*., by the ISP-provided information service presented in D1.1-Section 6. Using again the BitTorrent example, in this case, we assume that the tracker remains unchanged and replies a random list of peers to a peer's query. However, the originally querying peer can be given incentives, *e.g*., reduction of downloading time, in order to select only those peers from that list that are consistent with one or more criteria such as geographical proximity, RTT proximity, hops proximity, link cost, link congestion etc. This approach may be less effective than the previous one, but still can result to benefits for the user and the ISP.

Other example of this approach could be the QoS differentiation according to the end users' requests that is being defined in the ITU-T NGN or in ETSI/TISPAN. In this scenario, the end users' could request enhanced network performance for overlay based applications in order to optimise the different traffic profiles. *E.g*., Y.1541 ++ REFERENCE specifies the classes of services HighThroughputData and MMStreaming (it also specified other classes of services) that could benefit BitTorrent and Joost applications.

Effectively, in this scenario, the user could request enhanced capabilities for its overlay application. In this way, it could be possible to build carrier class overlay based applications that provides guarantees to the end users by just using the control plane being defined in convergent networks. Alternatively, the service provider could agree with ISP to manage its application related traffic with a specific class of services; such a case pertains to the non-transparent category of approaches.

3. Intervention of ISP to SOMs –Transparent ETM

In this case, the operator interferes in the overlay protocol and is responsible for the re-organization of the overlay network. This can be implemented by introducing extra equipment to its premises. For instance, the operator inserts new entities in the overlay network, *e.g*., caches or ISP-owned peers, which affect the overlay formation (see sections 4.2.3, 5 etc. of this deliverable). While the mechanisms described in Section 5 are somewhat artificial with respect to the SO aspect of the overlay, they may provide the possibility for a more direct influence for an ISP.

In this case, the underlay operator participates in the SOMs through these entities in order to achieve its own performance and cost optimization, *e.g*., reduction of resources consumption, reduction of inter-domain traffic, reduction of monetary cost paid to other operators (transit agreements). There are examples where an ISP just considers its own advantage, *e.g*., by throttling P2P traffic, regardless of the wishes of its customers.

However, the operator must also ensure that the end-users' performance does not degrade; otherwise he will end up losing customers. Here, the interaction of ETM and SOMs results in a win- non-lose situation, because it is enough if the end-users' performance remains the same. In addition, when intervention to SOMs is performed by the operator, all necessary underlay information is directly available by the operator himself. ETM, also by the operator, is necessary in order to concentrate and organize this information. Again, in the case of BitTorrent, we can assume that the ISP inserts an additional peer, with high upload bandwidth, which can be selected more frequently by other peers in the same autonomous system than the "regular" peers, thus leading to a decrease in the traffic of the inter-ISP link.

Having outlined the three possible scenarios of interworking between SOMs and ETM, we now turn our attention to several issues applicable to these interactions. First, we discuss the issue of timescales in which these interactions are effective. In particular, in the management plane, SOMs may be running in short timescales while ETM seems to be more of an outer loop in the sense that it has a strategic longer-term optimization objective. While information should be fed to the SOMs in shorter timescales by the ETM, so that the SOM is able to perform its actions, the underlay short-term actions may influence noticeably the overlay on longer timescales.

In addition, the interaction between ETM and SOMs may create a monitoring/accounting overhead, so, information feedback should be done efficiently in order to provide only information that is not already available to the SOM.

An issue that arises here is that it is not yet obvious whether the achievement of win-win-win situation requires truthful provision of underlay information to SOMs by the ETM. Likewise, it is not clear whether malicious use of such information can be beneficial to entities in the overlay. For example, in the case of BitTorrent, the fact that the tracker gains access to provider-specific information leads to a discussion of the risks introduced by a potentially malicious tracker. Moreover, an issue arises whether the ISP can benefit even further by not providing the information of the actual situation in the underlay. Apart from that, the fact that the benefits described in this example can be achieved has of course to be shown in this project, at least in certain cases. It may be that unforeseen side effects occur, *e.g.*, a selfish end-user may find a way to exploit the information provided by an ISP in order to achieve an even better performance, negating the advantage of the ISP in the process. Also, negative effects on the stability and load distribution in the overlay network should be avoided.

The potential benefits that can be achieved by employing ETM-aware SOMs differ between the aforementioned search and distribution overlays. Since distribution overlays generate more traffic, the savings potential by influencing this traffic is higher. Also, the content the user is interested in is transferred using a distribution overlay, which means that SOMs influencing the efficiency of these overlays can improve the end users QoS/QoE. This may pose a provider with opportunities to offer new services. For larger content, such as video files or streams, the time to locate the content is much shorter than the time it takes to transfer the content. While long answer times in a search are also annoying to end users, the search only constitutes a very small part of the whole overlay usage scenario. Therefore, a first tentative preference of SOMs in distribution overlays may be expressed with respect to SmoothIT. However, since more than one SOM might be applied to the applications selected for trials, it is too early to make a selection.

Last but not least, each SOM performs myopic individual optimization (its own net benefit), but it is still unclear whether this leads to social optimization (social welfare). That is, it is not clear whether incentive compatibility applies. Each agent in SOMs performs myopic individual optimization. For example, locality aware selection in BitTorrent can influence adversely the overall distribution of content, which in turn may influence the adversely the overall performance of end users.

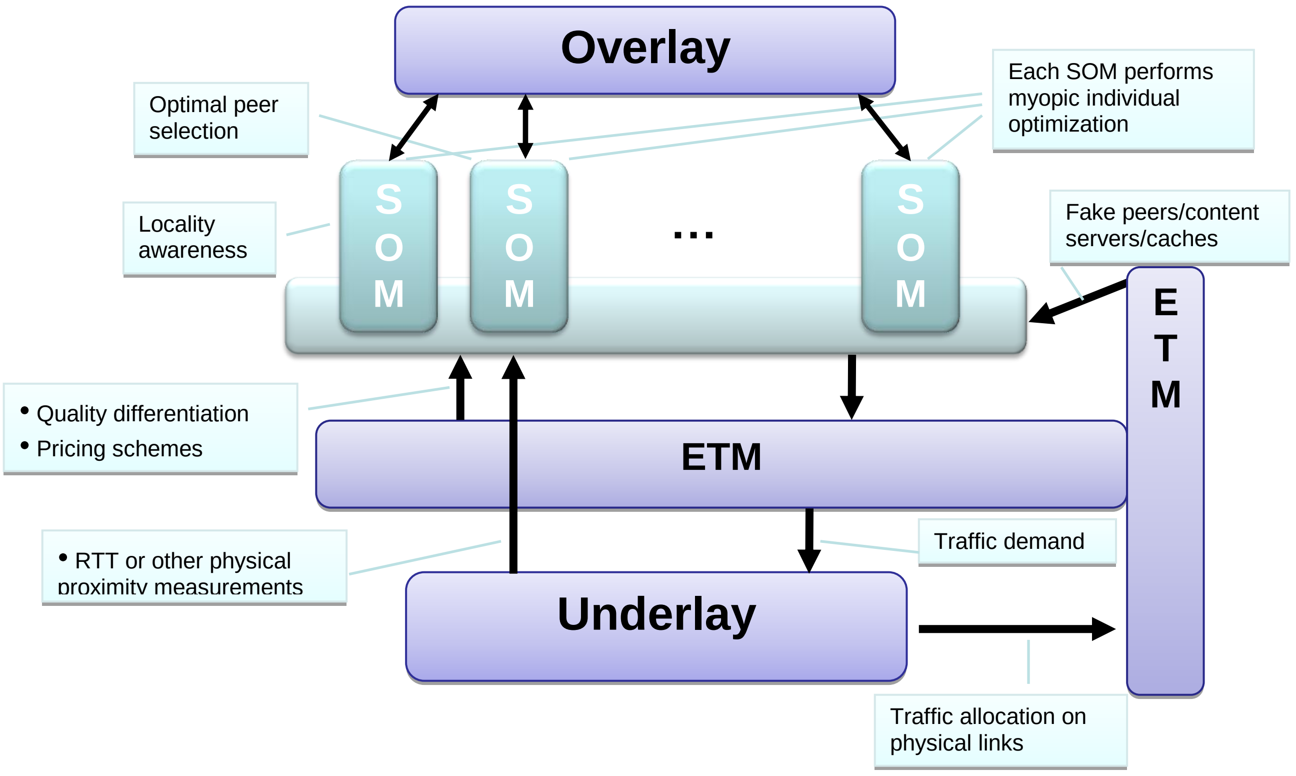

Overlay
Optimal peer selection
Each SOM performs myopic individual optimization
Locality awareness
SOM
SOM
...
SOM
Fake peers/content servers/caches
ETM
Quality differentiation
Pricing schemes
ETM
RTT or other physical proximity measurements
Traffic demand
Underlay
Traffic allocation on physical links

Figure 7. Graphical representation of possible SOM and ETM interactions

Figure 8 summarizes all possible ways of interworking between ETM and SOMs. Also, the information that is exchanged between the various layers is depicted.

7 Summary and Conclusions

Numerous Self-Organization Mechanisms (SOMs) for overlays exist. We can discern algorithms that work predominantly for search overlays, while others are only applicable to distribution overlays. Many mechanisms of both classes try to adapt the overlay to the underlay conditions, based on available information like Round-Trip Times or available bandwidth. However, this information is generated at the peers, i.e., at the end systems of the underlay.

This is the starting point for influencing the overlay as planned in the SmoothIT project. With different and more detailed information available to the peers, SOMs should be able to form an overlay and shape its traffic according to the aims of Economic Traffic Management (ETM). Ideally, this results in advantages for all involved parties, such as end users and ISPs, in comparison to a standard overlay implementation.

In this deliverable, we presented an overview on overlay architectures, SOMs employed therein and the metrics used to steer these mechanisms. We considered search and distribution overlays, and the neighbor selection and resource exchange mechanisms they use. Additionally, we presented methods to monitor vital values of an overlay. While the main focus is on self-organization, we also included architectural changes that are not self-organizing in the strict sense, but that still allow for influence on an overlay.

The interplay between all these mechanisms and the original aim of the project, i.e., to enable ETM in overlays, was described and illustrated. The examples given are possible implementations of these principles, but do not constitute the solution evaluated in later stages of this project. While it was not the aim of this document to make a selection from the presented mechanisms, the parallel discussion contained in D1.1 shows that distribution overlays and the associated SOM probably offer a higher potential for improvement. Where applicable, the content presented is rated with respect to the application selected in D1.1, Section 7.3, and to the future project phases. Based on this, we view the different underlay metrics as well as the resource exchange mechanisms as the most relevant aspects of self-organization. Additional mechanisms, such as monitoring or pricing, may however still be of interest to support ETM. These findings are summarized in Table 3.

Table 3. Summary of SOMs and their relevance for the project

Mechanism class	Field of application	Relevance for SmoothIT	Reason
Metrics	Peer selection	Important/will be used (especially underlay metrics)	The SIS will use and combine metrics to rank peers
Neighbor Selection	DHTs	Little importance	Considered application is no DHT
Peer Selection	P2P CDNs	Important/will be used	These mechanisms govern where data traffic flows
Chunk Selection	P2P CDNs	Current mechanisms may suffice, will however be part of the implementation	Does not directly influence traffic flows
Data replication/ caching	All overlay networks	Unclear	Depends on the influence of SmoothIT changes on the overlays
Monitoring	Mainly DHTs	Little importance	Considered application is no DHT, general concepts may be of interest
Caches, Proxies, Crawlers, IoP	P2P CDNs	Unclear	Require heavy intervention, but may help as additional support
Path Changes, Pricing	P2P CDNs	Relevant as incentives	May be able to provide users with incentives

8 References

[AD01] K. Aberer and Z. Despotovic: *Managing Trust in a Peer-to-Peer Information System*; In Proc. of the 10th International Conference on Information and Knowledge Management, New York, November 2001.

[AFS07] Vinay Aggarwal, Anja Feldmann, Christian Scheideler: *Can ISPS and P2P users cooperate for improved performance?;* SIGCOMM Computer Communication Review, Volume 37, Number 3, Pages 29-40, New York, USA, 2007.

[BHG87] P. Bernstein, V. Hadzilacos, N. Goodman: *Concurrency Control and Recovery in Database Systems*; Massachusetts, Addison-Wesley Publishers.

[BL07] A. Binzenhöfer, K. Leibnitz: *Estimating Churn in Structured P2P Networks*; 20th International Teletraffic Congress (ITC20), Ottawa, Canada, June 2007.

[BMS+02] R. Bhagwan, D. Moore, S. Savage, G. Voelker: *Replication Strategies for Highly Available Peer-to-Peer Storage*; FuDiCO: Future Directions in Distributed Computing, May 2002.

[BPL+02] E. Balafoutis, A. Panagakis, N. Laoutaris, I. Stavrakakis: *The impact of replacement granularity on video caching*; IFIP Networking 2002.

[BS07] A. Binzenhöfer, H. Schnabel: *Improving the Performance and Robustness of Kademlia-based Overlay Networks*; KIVS 2007, Bern, Switzerland, February 2007.

[BSH05] A. Binzenhöfer, D. Staehle, R. Henjes: *On the Fly Estimation of the Peer Population in a Chord-based P2P System*; 19th International Teletraffic Congress (ITC19), Beijing, China, September 2005.

[CDH+02] M. Castro, P. Druschel, Y.C. Hu, A. Rowstron: *Topology-aware routing in structured peer-to-peer overlay networks*; Technical Report, Microsoft Research, MSR-TR-2002-82, 2002.

[CDK+03] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, A. Singh: *SplitStream: High-Bandwidth Multicast in Cooperative Environments*; SOSP'03, New York, U.S.A., October 2003.

[CES+91] R. Cocci, D. Estrin, S. Shenker, and L. Zhang: *A Study of Priority Pricing in Multiple Service Class Networks*; In Proc. of the ACM Conference on Communications Architecture & Protocols (SIGCOMM '91), Zurich, Switzerland, September 3-6, 1991.

[Ci06] Cisco Systems: *Cisco IOS Netflow: Introduction*; http://www.cisco.com/en/US/products/ps6601/prod-ucts_ios_protocol_group_home.html, May 2006.

[CLG+03] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko: *Diameter Base Protocol*; Internet Engineering Task Force (IETF) RFC 3588, September 2003.

[CM96] A. Chavez and P. Maes, *Kasbah: An Agent Marketplace for Buying and Selling Goods*; 1st International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, U.K., April 1996, pp 75-90.

[DCK+04] F. Dabek, R. Cox, F. Kaashoek, R. Morris: *Vivaldi: a decentralized network coordinate system*; SIGCOMM 2004.

[DCP+03] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi and P. Samarati: *Managing and Sharing Servents' Reputations in P2P Systems*; IEEE Transactions on Knowledge and Data Engineering, vol. 15, n.4, July/Aug.ust 2003, pp. 840-854.

[De03] C. Dellarocas: *The Digitization of Word-of-Mouth: Promise and Challenges of Online Feedback Mechanisms*; Management Science, October 2003.

[EHB+07] K. Eger, T. Hoßfeld, A. Binzenhöfer, G. Kunzmann: *Efficient Simulation of Large-Scale P2P Networks: Packet-level vs. Flow-level Simulation*; 2nd Workshop on the Use of P2P, GRID and Agents for the Development of Content Networks (UPGRADE-CN'07) in conjunction with the 16th IEEE HPDC, Monterey Bay, USA, June 2007.

[FDL00] M. Falkner, M. Devetsikiotis, and I. Lambadaris: *An Overview of Pricing Concepts for Broadband IP Networks*; IEEE Communications Surveys & Tutorials, Vol. 3(2):2–13, Second Quarter, 2000.

[FLS+04] M. Feldman, K. Lai, I. Stoica and J. Chuang: *Robust Incentive Techniques for Peer-to-Peer Networks*; In Proc. of the 4th ACM Conference on Electronic Commerce, New York, NY, USA, May 2004.

[GB06] S. Goel, R. Buyya: *Data Replication Strategies in Wide Area Distributed Systems*; in Enterprise Service Computing: From Concept to Deployment, pp. 211-241, Hershey, 2006. [HFP+05] M. Howarth, P. Flegkas, G. Pavlou, N. Wang, P. Trimintzios, D. Griffin, J. Griem, M. Boucadair, P. Morand, H. Asgari and P. Georgatsos: *Provisioning for Inter-domain Quality of Service: The MESCAL Approach*; IEEE Communications Magazine, Vol. 43, No. 6, June 2005, pp 129-137.

[GES05] P Garbacki, D Epema, M van Steen: *Two-level semantic caching scheme for super-peer networks*; 10th International Workshop on Web Content Caching and Distribution, Sophia Antipolis, 2005. [HMT+05] T. Hoßfeld, A. Mäder, K. Tutschku, P. Tran-Gia, F.-U. Andersen, H. deMeer, I. Dedinski: *Comparison of Crawling Strategies for an Optimized Mobile P2P Architecture*; 19th International Teletraffic Congress (ITC19), Beijing, China, September 2005.

[GGG+03] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, I. Stoica: *The Impact of DHT Routing Geometry on Resilience and Proximity*; In Proceedings of ACM SIGCOMM 2003, March, 2003.

[GLL07] H. Guo, K. Lo, J. Li: *An Efficient Caching Scheme for On-demand Streaming Service on Overlay Networks*; CCNC 2007.

[Ha68] G. Hardin: *The Tragedy of the Commons*; Science, volume 162, pp. 1243-1248, 1968.

[HAS+07] M. Hosseini, D.T. Ahmed, S. Shirmohammadi, N.D. Georganas: *A Survey of Application-Layer Multicast Protocols*; IEEE Communications Surveys & Tutorials, Vol. 9, Issue 3, pp. 58-74, 2007.

[HOT+06] T. Hoßfeld, S. Oechsner, K. Tutschku, F.-U. Andersen, L. Caviglione: *Supporting Vertical Handover by Using a Pastry Peer-to-Peer Overlay Network*; Mobile Peer-to-Peer Computing MP2P'06, in conjunction with the 4th IEEE International Conference on Pervasive Computing and Communications (PerCom'06), Pisa, Italy, March 2006.

[HKF+04] S. Handurukande, A.-M. Kermarrec, F. Le Fessant, L. Massoulié: *Exploiting Semantic Clustering in the eDonkey P2P Network*; ACM Sigops 11, Leuven, Belgium, September 2004.

[HW04] J. Hwang, C. H. Lee: *Agent-based Modeling for Differentiated Admission in P2P Systems Using Evolutionary Game Theory Focused on Ownership Reputation*; in Proc. of the Workshop on Economics of Peer-to-Peer Systems, Harvard, MA, June 2004.

[JHF03] A. Josang, S. Hird, E. Faccer: *Simulating the Effect of Reputation Systems on e-Markets*; in Proc. of the 1st International Conference on Trust Management, Crete, Greece, May 2003.

[KSG03] S. D. Kamvar, M. T. Schlosser and H. Garcia-Molina: *EigenRep: Reputation Management in peer-to-peer Networks*; In Proc. of the Twelfth International World Wide Web Conference, Budapest, Hungary, May 2003.

[M00] *M3I: Market Managed Multi-service Internet*; 5th Framework EU Project, IST Program, No. 11429, URL: http://www.m3i.org, January 2000.

[MG04] S. Marti and H. Garcia-Molina: *Limited Reputation Sharing in P2P Systems*; In Proc. of the ACM conference on Electronic commerce, New York, NY, USA, May 2004.

[MLL+04] T. B. Ma, S. C. M. Lee, J. C. S. Lui, D. K. Y. Yau: *A Game Theoretic Approach to Provide Incentive and Service Differentiation in P2P Networks*; in Proc. of ACM SIGMETRICS/PERFORMANCE, June 2004.

[MM02] P. Maymounkov, D. Mazieres: *Kademlia: A Peer-to-Peer Information System Based on the XOR Metric*; International workshop on Peer-To-Peer Systems, October 2002.

[MM08] MaxMind, Inc.: *www.maxmind.com*; May 2008.

[MMK+06] L. Massoulie, E. Le Merrer, A. Kermarrec and A. Ganesh: *Peer Counting and Sampling in Overlay Networks: Random Walk Methods*; PODC’06, Denver, Colorado, July 2006.

[NWD03] T.-W. J. Ngan, D. S. Wallach, and P. Druschel: *Enforcing Fair Sharing of Peer-to-Peer Resources*; In Proc. of the 2nd International Workshop on Peer-to-Peer Systems, Berkeley, California, February 2003.

[OAM+05] J. Oberender, F.-U. Andersen, H. deMeer, I. Dedinski, T. Hoßfeld, C. Kappler, A. Mäder, K. Tutschku: *Enabling Mobile Peer-to-Peer Networking; in Mobile and Wireless Systems*; Dagstuhl, Germany, January 2005.

[PKS00] G. Pierre, I. Kuz, M. van Steen: *Adaptive Replicated Web Documents*; Technical Report IR-477, September 2000.

[PP08] The P4P project: *http://www.openp4p.net/*; May 2008.

[PS05] *Enforcing Truthful-Rating Equilibria in Electronic Marketplaces*; In Proc. of IEEE ICDCS Workshop on Incentive-Based Computing, Lisbon, Portugal, July 2006.

[PS06] T. G. Papaioannou and G. D. Stamoulis: *Reputation-based Policies that Provide the Right Incentives in Peer-to-Peer Environments*; Computer Networks (Special Issue on Management in Peer-to-Peer Systems: Trust, Reputation and Security), Elsevier, vol. 50, issue 4, pp. 563-578, 2006.

[RD01] A. Rowstron, P. Druschel: *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems*; In Lecture Notes in Computer Science, vol. 2218, pp 329ff, 2001.

[RHS03] P. Reichl, D. Hausheer, and B. Stiller: *The Cumulus Pricing Model as an Adaptive Framework for Feasible, Efficient, and User-Friendly Tariffing of Internet Services*; Computer Networks, Vol. 43(1):3–24, September 2003.

[RFH+01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker: *A scalable content-addressable network*; SIGCOMM, 2001.

[RL05] R. Rodrigues, B. Liskov: *High Availability in DHTs: Erasure Coding vs. Replication*; 4th International Workshop, IPTPS 2005, Ithaca, NY, USA, February 24-25, 2005.

[SFR00] M. Schillo, P. Funk and M. Rovatsos: *Using Trust for Detecting Deceitful Agents in Artificial Societies*; Applied Artificial Intelligence, 14:825-848, 2000.

[SGG02] S. Saroiu, P. Gummadi, S. Gribble: *A measurement study of peer-to-peer file sharing systems*; MMCN, 2002.

[SI08] The SmoothIT project: *Deliverable 1.1: Requirements and Application Classes and Traffic Characteristics;* June 2008.

[SML+02] I. Stoica, R. Morris, D. Liven-Nowell, D. Karger, M. F. Kaashoek, F. Dabek H. Balakrishnan: *Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications*; Technical Report, MIT Laboratory for Computer Science, MIT-LCS-TR-819, January 2002.

[TT02] M. Tran, W. Tavanapong: *An overlay caching scheme for overlay network*; in Technical Report TR02-11 Department of Computer Science, October 2002.

[Tu05] K. Tutschku (ed.): *First report on future control architectures for NGI services*; Euro-NGI deliverable D.JRA.1.4.2, May 2005.

[VIF06] Aggelos Vlavianos, Marios Iliofotou, Michalis Faloutsos: *BiToS: Enhancing BitTorrent for Supporting Streaming Applications*; INFOCOM 2006. 25th IEEE International Conference on Computer Communications, Proceedings, pp.1-6, April 2006.

[W99] J. Wang: *A Survey of Web Caching Schemes for the Internet*; ACM Computer Communication Review, 29(5):36–46, October 1999.

[WGR05] K. Wehrle, S. Götz, S. Rieche: *Distributed Hash Tables*; in: R. Steinmetz, K. Wehrle (eds.): Peer-to-peer Systems and Applications, LNCS 3485, Springer Verlag, Berlin Heidelberg 2005, chapter 7, pp.79-93.

[YS02] B. Yu and M. P. Singh: *Distributed Reputation Management for Electronic Commerce*; Computational Intelligence, Vol. 18, Issue 4, pp. 535-549, 2002.

[ZDK06] S. Zoels, Z. Despotovic, W. Kellerer: *Cost-Based Analysis of Hierarchical DHT Design*; IEEE International Conference on Peer-to-Peer Computing.

[ZDK07] S. Zoels, Z. Despotovic, W. Kellerer: *Load Balancing in a Hierarchical DHT-based P2P System*; International Conference on Collaborative Computing, 2007.

[ZDK08] S. Zoels, Z. Despotovic, W. Kellerer: *On Hierarchical DHT Systems - An Analytical Approach for Optimal Designs*; Elsevier Journal of Computer Communication - Special Issue: Disruptive Networking with Peer-to-Peer Systems 31(3) (2008) 576–590.

9 Terms and Definitions

Overlay	A logical network
Overlay structure	The graph created by interpreting the peers as nodes and the logical connections between them as edges. For structured overlays, also the underlying organizing principle, *e.g.*, ring, d-torus, etc.
Underlay	The physical network, normally the internet or parts thereof
Client	The software installed on a users machine, implementing the local application and peer functionality
Application	The functionality the user wants to use, *e.g.*, file download, watching video, phone
User	The person using an application
QoE	Quality of Experience; A subjective measure of performance in a system
ISP, underlay provider, operator	The owner of a physical network, offering services for and via that network
Overlay provider	If existent, the party providing necessary entities for an overlay to work
Peer, Node	One entity that together with all other peers/nodes builds an overlay by establishing connections between them

10 Abbreviations

3GPP	3rd Generation Partnership Project
AAA	Authentication, Authorization, Accounting
ALM	Application Layer Multicast
AS	Autonomous System
BGP	Border Gateway Protocol
CAN	Content-Addressable Network
DHT	Distributed Hash Table
DSL	Digital Subscriber Line
ETM	Economic Traffic Management

GPRS	General Packet Radio Service
HTML	Hypertext Markup Language
ICMP	Internet Control Message Protocol
ID	Identifier
IP	Internet Protocol
ISP	Internet Service Provider
LRU	Least Recently Used
MSD	Multiple-Source Download
OCS	Overlay Caching Scheme
P2P	Peer-to-Peer
PC	Personal Computer
PFSP	Partial File Sharing Protocol
QoE	Quality of Experience
QoS	Quality of Service
RTT	Round-Trip Time
SmoothIT	Simple Economic Management Approaches of Overlay Traffic in Heterogeneous Internet Topologies
SO	Self-Organization
SOM	Self-Organization Mechanism
STREP	Specific Targeted Research Project
TTL	Time to Live
UMTS	Universal Mobile Telecommunication System
VoD	Video-on-Demand
WLAN	Wireless Local Area Network

11 Acknowledgements

This deliverable was made possible due to the large and open help of the WP2 team of the SmoothIT team within this STREP. The authors would like to thank especially the reviewers Peter Racz (UZH) and Spiros Spirou (ICOM) for their valuable comments.