Project Number:	**Contract Number: INFSO-ICT-224287**
Project acronym:	**VITAL++**
Project Title:	**Embedding P2P Technology in Next Generation Networks: A New Communication Paradigm & Experimentation Infrastructure**
Title of Report	**Content based services**

Instrument:	STREP
Theme:	ICT-2-1.6
Report Due:	M21
Report Delivered:	
Lead Contractor for this deliverable:	RBB
Contributors to this deliverable:	Nicolas de Abreu Pereira (RBB), Bjoern Stockleben (RBB), Shane Dempsey (WIT), Stephen Garvey (WIT), Charalabos Skianis (UoP), Nikolaos Efthymiopoulos (UoP), Kostas Koutsopoulos (BCT), David Florez Rodriguez (TID), Juana Sanchez (TID).
Estimated person Months:	33
Start date of project:	1st June 2008
Project duration	30 months
Revision:	Version 1.4
Dissemination Level:	PU - Public

This page intentionally blank

Table of Contents

List of Figures

List of Tables

1 Executive summary

This deliverable aims to describe the content services developed and deployed in the course of the project and the main technical components and processes that are required to deploy these content services.

This document distinguishes between content services and content-related services.

The first part of the document (#2) describes in details the major content-related services developed in the course of VITAL++. These descriptions are kept generic, aiming to explain the components' objectives and functionalities.

The second part (#3) describes a number of Content Services which could be deployed in the VITAL++ architecture making use of the components described in chapter #2. This chapter focuses pre-dominantly on the VITAL++ demonstration service which incorporates most of the features and functions provided by the VITAL++ architecture, the SoftMix Scenario, which is described in #3.2. Eventually, #3.2.4 describes how the Content-related Services in #2 are being integrated and used in the demonstration scenario, detailing the major use case scenarios that occur in the SoftMix Service.

2 Content-related Components

This chapter focuses on the main components that have been developed or adapted in VITAL++ to cater for the content-based requirements of the envisaged content services, enabling the respective processes. The first part (#2.1) lays out the content distribution mechanisms used in VITAL++ (Live Streaming, Content on Demand, and File Sharing), the second part (#2.2) describes the transcoding components and mechanisms that enable device-specific content provision and the last part (#2.3) details the processes behind Conditional Access Management, including content protection and billing procedures.

2.1 Content Distribution Types

We specify here three types of content distributions that categorize technically the applications that we envisage:

1. Live Streaming
2. Content on Demand
3. File Sharing

In file sharing users can publish and download files. This service implements two functionalitites that users are able to use. They can publish files and they can request files. Each file is visible to the users as soon as it was published and uploaded to a VITAL++ server. In this category each user is able to request a file and it is available after the completion of its download. Users cannot use a file while it is being downloaded but have to wait for the download to be completed.

The second category is P2P live streaming. In this category a video/audio stream is produced in real time with a service rate of μ bits/sec. Every second of the stream is then divided into N_b blocks. So each block is generated every $1/N_b$ seconds with a size equal to $L_b=\mu/N_b$ bits. These blocks are sent directly to a small subset of peers. Consequently, peers that are neighbours in the overlay mutually dynamically exchange blocks until they acquire the whole stream. This exchange is carried out according to a block scheduler that runs in every peer and maintains a buffer of all N_b*t_s blocks generated within a sliding window of t_s seconds (with t_S we denote the setup time).

The third category, content on demand, is similar to live streaming but in this case the content is not broadcast to every participating user but each user is able to request each multimedia object every time instant. Again every second of the stream is then divided into N_b blocks. So each block is generated every $1/N_b$ seconds with a size equal to $L_b=\mu/N_b$ bits. In case that our system finds dependencies between peers in terms of multimedia objects that they consume forms an overlay and peers contribute their upload bandwidth.

2.1.1 Live Streaming

For multimedia broadcast over P2P or even just multicast of content which is of interest for many users in a P2P network, Live Streaming is the best technical solution. The crucial aim in terms of technical development is the creation of an algorithm with which participating peers will be able to exchange blocks and have in real time optimal and stable multimedia file distribution.

In order to achieve fair block distribution and build a system that is adaptable to dynamic peer behavior and network conditions, the selection of the receiving peer is the responsibility of the sending peer. This selection is taking place exactly before the beginning of the transmission of a block. On the other hand, the receiving peer proactively notifies candidate sending peers about the block that it wishes to receive. In this way, we can achieve fast and complete diffusion of each block while we avoid duplicate block transmissions from different sending peers to the same receiving peer. We consider this receiver driven block selection approach as the most efficient one in distributing blocks since the receiving peer always has a better knowledge about the rarity of its missing blocks in the buffers of its neighbours and this knowledge can be communicated to its neighbours that they act as sending peers. Then the problem of block distribution has been shifted toward coordinating these sending peers in a distributed manner such that they avoid duplicate block transmissions and prioritize the transmission of rare blocks in a neighbourhood.

Furthermore, as the rate of the requests by the receiving peers has to be kept at least equal to the rate at which blocks of the stream are consumed for the video playback, the receiving peer sends its block requests only to those candidate sending peers that can meet this constraint, namely those who have sufficient uploading capabilities. Towards this goal, each peer implicitly announces its serving capability by periodically issuing tokens to a set of peers with size equal to its uploading capabilities. These tokens have to be distributed uniformly to the participating peers in order to request blocks and eventually acquire the video stream.

Taking into account the objectives above, the VITAL++ scheduler is composed of three components. The first is the **token generation algorithm** where each potential sender periodically defines the set of the potential receivers that it is able to serve and issues tokens to them. The second is the **proactive block request algorithm** where a potential receiver matches the tokens that it has received with different blocks that it misses from their buffers. We note the period in which these two algorithms are executed, request_interval. The third component is the **neighbour selection algorithm**, which is executed by the sending peer exactly before the transmission of a block taking into account the requests from receiving peers, their upload bandwidth capabilities and the amount of blocks that they miss.

2.1.1.1 Token Generation Algorithm

Each peer i periodically executes an algorithm that selects a subset of its neighbours that can be served according to its available uploading bandwidth capabilities c(i) which are dynamically measured by the peer. The selection process is described later in this section. Denoting this set, token_set(i), the algorithm calculates its size according to the following formula:

$$|\text{token\_set}(i)| = c(i)*\text{request\_interval}/L_b$$

Super peers that participate in the super peer overlay have their available uploading capabilities greater than the video serving rate, i.e. $c(i)>s$. Therefore, the rate of the generated tokens per peer is $s*\text{request\_interval}/L_b$ which is equal to $N_b*\text{request\_interval}$ which is enough for the distribution of the stream in the super peer overlay. Similarly, the number of tokens that each super peer issues to the base overlay is proportional to its excess bandwidth (BE) as defined in D3.1 in the overlay management section and equal to $BE(i)*\text{request\_interval}/L_b$.

In contrast, slow peers issue all of their tokens only to the base overlay. In this way we avoid the transmission of a block from a slow peer to a super peer because experience shows that the fast diffusion of the blocks in the super peer overlay is critical for the reduction of the setup time of the system.

A key requirement to have a P2P live streaming system with high bandwidth utilization and timely stream distribution is to uniformly distribute the sum of these tokens to every participating peer since every one of them has to acquire blocks with a rate equal to N_b. We denote the probability in which a sending peer i selects a peer j in the token_set(i) as P(i,j). Initially, the probabilities for each super peer i are assigned as $P(i,j)= 1/M_S$ for each j that is also a super peer, and $P(i,j)= 1/M_I$ for each j that is a slow peer. Finally, the probabilities for each slow peer i are $P(i,j)= 1/M_B$.[1]

[1] These parameters are also parameters in the overlay and they are analyzed in D3.1 in section 8.3.

Using these probability assignments by default, leads to a distribution of tokens among all peers that follows the normal distribution. This implies that some receiving nodes will get more tokens than others that are deprived of them. Therefore, we need to introduce a mechanism that imposes a distribution of tokens in such a way that all receiving peers eventually get the same number of tokens. In order to achieve this, each time that this algorithm is executed it examines whether the potential receiver that belongs to the token_set(i) - i.e. selected by the sending node - made a block request in the last request_interval. If the receiving peer j has indeed issued a request to peer i, the latter immediately increases P(i,j) by a fixed percentage called per. At the same time, it decreases this probability according to per. As a result, receiving peers that made block requests continue to be served by specific sending nodes, while sending nodes select new peers for receiving nodes in the token_set(i) since the probability of the previously selected peers have been decreased due to the fact that they didn't request any block. We note here that there is normalization of all probabilities after each recalculation of probabilities. This includes all peers that are neighbours and they may not belong to the token_set(i).

In order to make sure that the increase or decrease of probabilities does not go towards one or zero respectively, we also put an upper or lower bound. Every time these probabilities hit any of these bounds they stay there until there is a decrease or increase respectively.

2.1.1.2 Proactive Block Assignment and Request

The second algorithm complements the previous by proactively determining which block a peer should request from those neighbours that have already sent a token to it during the last request_interval.

The block assignment process is accomplished by performing a matching algorithm between the missing blocks and those neighbours that have them, requesting a different block from each neighbour, in such a way as to maximize the number of blocks that it can request. As a result duplicate block transmissions are greatly reduced and content bottleneck is avoided as the newly produced and rare blocks have a high probability to be requested in order to maximize the number of the requested blocks.

Each peer i acting as a potential block receiver from its neighbours performs the matching algorithm every time that receives a new block. The output of the algorithm is propagated immediately to its neighbours. Each neighbour j of peer i has a set of blocks that we note as S_j we also define a parameter c(j.k) that is 1 if peer j has block k and 0 if peer j miss block k. We additionally note the all the neighbours of a peer i as Neigh(i) and we define a parameter a(i,j,k) that represents the output of the matching algorithm. We allow this parameter to take two values 0 and 1 and in the second case peer i requests from j the block k. The algorithm solves a linear optimization problem that is:

$$max \sum_{j \in Neigh(i)} \sum_{k \in Sj, k \ni Si} a(i,j,k) * c(j,k) \qquad (1)$$

Constrained to for each block k:

$$\sum_{j \in Neigh(i)} a(i,j,k) \leq 1 \qquad (2)$$

And for each neigbor j that belongs to Neigh(i):

$$\sum_{k \in Sj} a(i,j,k) \leq 1 \qquad (3)$$

Intuitively the maximization function ensures that each peer will request as many blocks as possible that are available in the buffers of its neighbours. As we have observed this function seeds up the diffusion of rare or newly produced blocks than randomized request. Furthermore it maximizes the total number of blocks that will be requested and in this way we minimize the probability to have idle upload bandwidths in Neigh(i). For each block we introduce a constraint that prevents the request of the same block to two potential senders and in this way we almost eliminate duplicate block transmissions. Finally, we limit the number of requests at each neighbour to be equal to, or less than one in order to allow him distribute his upload bandwidth capabilities also to other potential receiver peers.

As it is observed in the evaluation section, our matching algorithms increases the diffusion speed of each block while it maximizes the upload bandwidth utilization of the participating peers by avoiding content bottleneck and duplicate block transmissions.

2.1.1.3 Neighbour Selection for block transmission

Our neighbour selection algorithm takes into account two objectives: the first is the equal percentage of block receptions in every node and the second is the preference towards nodes of high uploading bandwidths in order to achieve fast stream distribution.

Towards this end we define a decision function, d(i,j) that provides a metric used by sender peer i for the selection of a neighbouring peer j for block transmission. The decision function is given by the following formula:

$$d(i,j) = \frac{difference(i,j)}{per * buf\_size} - \frac{rank(i,j)}{|neigbors(i)|} \qquad (4)$$

The node selected for block transmission is the one with the maximum d(i,j) □j that has issued a request to node i. In this equation the quantity difference expresses the number of blocks that the sender currently possesses and the receiver misses, |neighbours(i)| denotes the total number of neighbours of node i that have made a request to i. rank(i,j) is a function that returns the position of node j in a list with neighbours ordered in descending uploading bandwidth value. We have chosen to model the network bandwidths in this way in order to make our scheduler independent of the uploading bandwidth data set and as such suitable for every uploading bandwidth distribution. Moreover, buf_size is equal to N_b*t_s and denotes the number of blocks that nodes exchange at each time instant. Finally, the parameter per is a constant representing the percentage of the buffer size. We have successfully experimented with values of parameter per that are between 5%-10% of the buffer size.

If we examine the second term of the decision function, we note that it is a linear function of rank(i,j)assuming values in the range of [0,1], because 0<rank(i.j)<=|neighbours(i)|. When nodes have small differences for missing blocks, the first term is very small and so rank(i,j) has a dominating effect on the selection of node j and so the diffusion of blocks is done by favouring nodes with high upload bandwidth capabilities. On the other hand, when differences for missing blocks in the order of per*buf_size are observed, our scheduler approximates the most deprived scheduler behaviour with difference(i,j)becoming the dominant parameter for selecting node j. In this way we guarantee high degrees of fairness in the distribution of blocks.

The size of the request interval plays a crucial role in the performance of our system and is accountable for the presence or not of duplicate block transmissions. When a node i starts the transmission of a new block to node j, it takes time equal to STT(i,j) for that block to reach node j. If the next request interval for node j is during this time interval then node j may request the same block from another node leading to a duplicate packet transmission. On the other hand, if node i starts transmitting a block to node j, after node j issued its new requests but before node i receives them, then again that could result in a duplicate block transmission.

So the ratio of potential duplicate block transmissions to normal block transmissions depends on the STT values between nodes and to the size of the request interval. Large value of request interval leads to low volumes of potential duplicate block transmissions, thus increasing the maximum achievable service rate of our system. On the other hand this also leads to the increase of the setup time for a given service rate as the new available blocks are propagated to the peers in the system with slower rate.

2.1.2 Content on Demand

Generally, in our understanding, Content on Demand (CoD) would be a kind of time-shift streaming; users can receive content from peers as long as any one other peer receives the stream, too. It is not to be understood as a single stream out of a content archive.

In the case of the SoftMix Scenario, Content on Demand is to be understood as a very quick File Sharing solution in the sense that, potentially, every single user may receive individual content according to their profiles. The main difference between Content on Demand and File Sharing is that in CoD users are able to use files and/or multimedia objects before their completion. Therefore, a crucial aim in terms of technical development is the modelling of the time deadlines in which each of the multimedia objects and/or file blocks have to be delivered and the creation of a mechanism that delivers each block of the object that each user requests before this deadline.

Content on demand follows the same architecture as live streaming. The algorithms for the token generation and the proactive block assignment and request are identical. On the other hand in order to have a successful scheduling algorithm for content on demand we need to modify the neighbour selection function. Towards this we again define a decision function, d(i,j) that provides a metric used by sender peer i for the selection of a neighbouring peer j for block transmission. The decision function is given by the following formula:

$$d(i,j) = \frac{deprivacy\,(i)}{time\ remain} \qquad (5)$$

The objective according to the requirements of Content on Demand is to select for transmission the peer that requires the highest bit rate in order to receive the file before the deadline for the playback. This deadline is defined according to the application and represents the user requirements. In more detail we define as deprivacy (i) the number of blocks that peer i misses and as time remain the time interval between the time instant when the algorithm is executed and the time instant of the deadline.

2.1.3 File Sharing

In file sharing users who have data objects can publish them with the context index mechanism of VITAL++ and by the use of the existing scheduling mechanisms as in bit torrent these files can be exchanged between users with the utilization of their uploading bandwidth capabilities. VITAL++ will not contribute to the further development of these mechanisms. The presentation of this mechanism is analyzed in D2.3. On the other hand, through the creation of a network dependent and optimized graph we are able to boost the performance and the stability in file sharing in terms of downloading latency and file availability. We analyze this in D3.2. Additionally user access and accounting are new features in P2P file sharing which we analyze in D4.1.

2.1.4 Evaluation of scheduling architectures

For the evaluation of our P2P streaming system we have used the OPNET Modeler v.14 in order to test our proposed system under various underlying network topologies and conditions. Here we present its performance based on a real topology that is also used as a reference topology for the evaluation of other systems. However, similar performance has been observed with all topologies we have worked with. In order to model heterogeneous uploading bandwidth capabilities we set the uploading bandwidths equal to 4000 kbps (class 4), 1000 kbps (class 3), 384 kbps (class 2), and 128 kbps (class 1) corresponding to a distribution of 15%, 25%, 40%, and 20 % of peers. The average uploading bandwidth of this distribution is around 1030 kbps; this value will be used hereafter as the average uploading capacity of the system.

In order to demonstrate the performance of Intra-DOA and Inter-DOA algorithms, as we describe in D3.1 we form a randomly created overlay with 2000 nodes where $M_B=M_S=M_I=8$. Later in the evaluation we demonstrate the performance of our system under different values of these parameters. The service rate μ of the stream is equal to 95% (around 975 kbps) of the average uploading capacity.

In the graphs following we evaluate the performance of our scheduler under various scenarios. Firstly, in graphs 1, 2 and 3 we show the performance of our system under static conditions. We simulated a system with 2000 nodes in which we applied our DOA algorithms until their convergence and then started the streaming process which takes place for 50 seconds (larger values have no effect on the results). We set the values of N_b and t_s to 14 blocks and 2 sec respectively. The video streaming playback rate μ is set to the 95% of the average capacity. Finally, as in static conditions the STTs between neighbouring peers are very close to the minimum possible, we set the request interval to $2/N_b$ equal to 140 ms. We should note here that the value of the request interval has no correlation with the value of N_b, but as every peer sends its buffer to its neighbours $1/N_b$ seconds with this way we reduce the control overhead of our scheduler by piggy-packing the token and request messages within the buffer transmissions.

In Figure 1 we demonstrate the effectiveness of our scheduler by means of the CDF of successful block receptions from peers. Our system manages to successfully deliver a stream of a video streaming rate very close to the average uploading capacities of the peers within a very small setup time period. This exhibits the high degrees of bandwidth utilization that we can achieve.

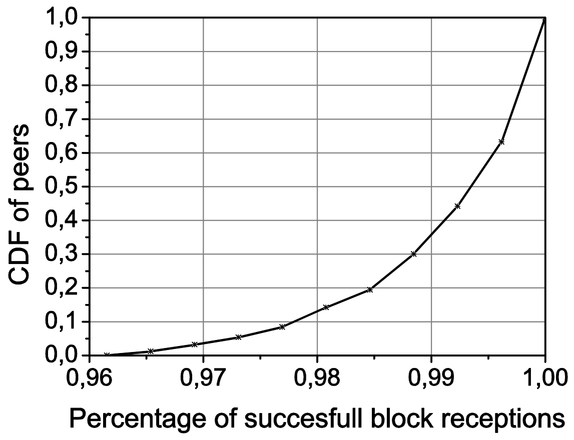


Figure 1: Percentage of the successful block receptions of nodes in static conditions.

In Figure 2 we show the CDF of the percentage of duplicate packets that was received by the peers. As we can observe our scheduler in conjunction with our locality aware overlay manages to reduce the percentage of duplicate packets to around only 1% percent. The percentage, which is translated to wasted bandwidth, along with the control overhead of our system, which has been taken into account on our simulations, give us an upper bound on the maximum achievable service rate for a given distribution of upload capacities.

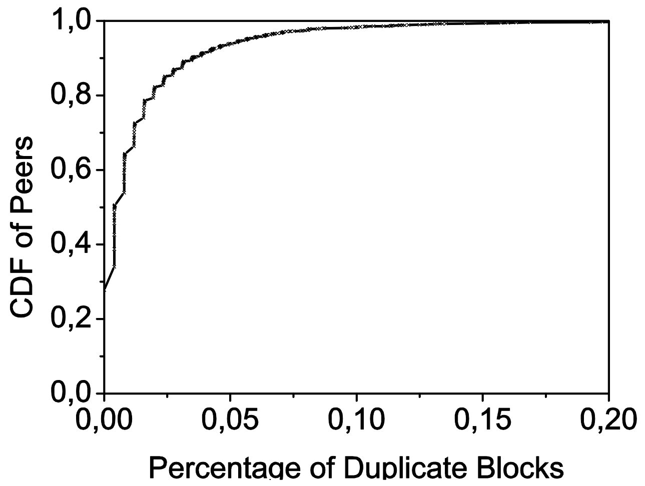


Figure 2: Duplicate block transmissions

Analytically, the control overhead for the overlay distributed optimization algorithms is negligible and less than 10 kbps per node as we have derived from our simulations. Furthermore, the token transmissions, the block requests and the buffer exchanges are all exchanged between neighbours with a control UDP packet periodically with a rate analogous to $1/N_b$. This length of the packet has always been less than 200 bits including the UDP and the IP headers. In addition, we have calculated each peer has on average 20 neighbours that exchange control packets with them. So the control overhead that our scheduler introduces is around 40 kbps which is 4% of the given average available uploading bandwidth. So the available useful bandwidth is around 95% of the average upload capacity and is the maximum achievable service rate as our system manages to utilize in an optimum way the upload capacities of the participating nodes.

Another proof of how our system manages to utilize the upload capacities is shown in Figure 3 which depicts the number of block requests that each peer received. We observe here that the number of block requests is almost identical for every class of peers and proportional to their uploading bandwidth. This shows the efficiency of the token generation and the proactive block request algorithms that manage to distribute the task of block propagation evenly to every peer according to its uploading capacity.

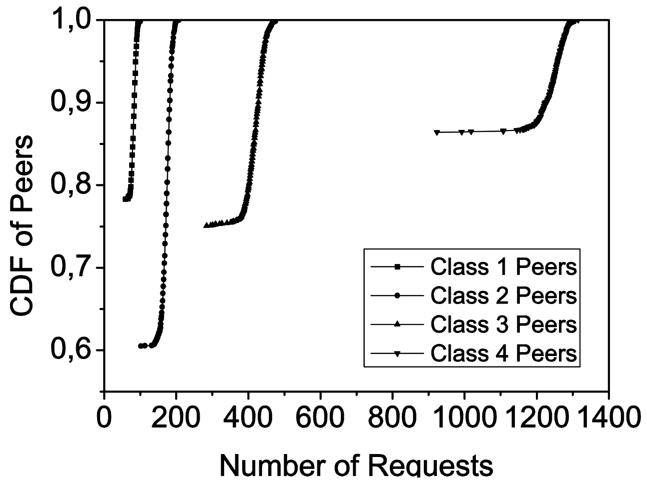


Figure 3: Number of requests for each Class of peers

In Figure 4 we demonstrate the behaviour of our system for different numbers of participating peers. As we observe our system's behaviour is almost independent of the number of the participating nodes indicating a potential asymptotic behaviour. This makes our system very stable and promising in terms of scalability.

In the rest of this section we evaluate our system under dynamic conditions. Again we simulate a system with 2000 nodes and we set the value of N_b to 14 and the service rate to 95% of the available uploading capacity. However, in order to make our system more stable in the presence of dynamic insertion and departures of peers, that have an impact on the energies of the participating peers, we have increased the `request_interval` from $2/N_b$ to $3/N_b$. This increase of the request interval has led to an increase of the setup time from 2 seconds to 3 seconds in order to have a complete diffusion of each block as we have analyzed in detail in section 3.4.

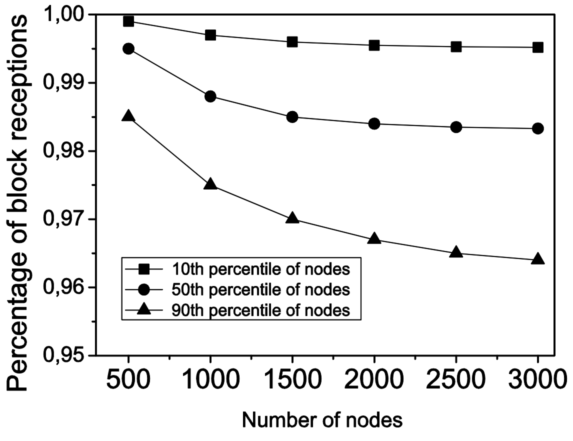


Figure 4: Performance as System Scales,

In Figure 5, we show the impact that the value of the `request_interval` has on the performance of our system. For the previous scenario we show the percentage of duplicate block reception for a request interval equal to $3/N_b$ and $2/N_b$ with setup time equal to 3 and 2 seconds respectively. As we expected, in the latter case the percentage is greater and above 5% meaning that the system can't deliver a stream of service rate equal to 95% of the available uploading bandwidth. If we want to deliver a stream with setup time equal to 2 seconds we need to degrade the service rate to about 87% of the available bandwidth. This shows that the value of the request interval is a trade-off between the maximum achievable service rate and minimum setup time.

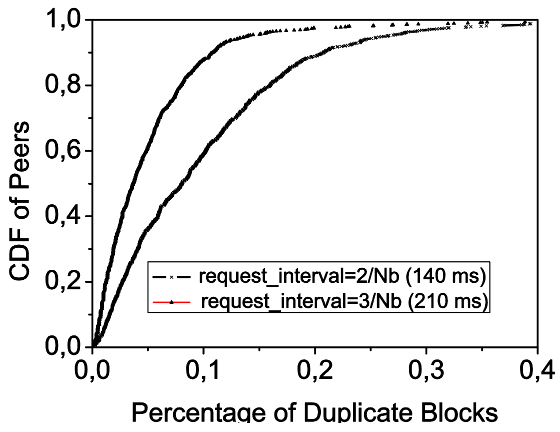


Figure 5: Duplicate blocks under variant request interval

In Figures 6 and 7 we demonstrate the percentage of successful block receptions and the percentage of duplicate block transmissions under variable block request frequency. The service rate equal to 95% of the available uploading bandwidth, the setup time is 2 sec and N_b is 14 blocks per second. From Figure 6 we can observe that there is an optimal request frequency that maximizes the percentage of successful block receptions. The explanation of this phenomenon is revealed in Figure 7 where we see a linear reduction of the percentage of duplicate block transmissions as the request frequency is reduced. On the other hand very small values of the request frequency lead to a condition where a sender has no blocks to transmit so that its uploading bandwidth remains idle. As we have observed through our evaluation scenarios the optimal request frequency is sensitive to the setup time and the network latencies between participating peers. On the other hand we observe that there is an interval in the request frequency from around 4 to 7 where we have a stable behaviour in the performance of the scheduler and this makes us optimistic for the stability of our system although we are not in position to calculate analytically the optimal request interval.

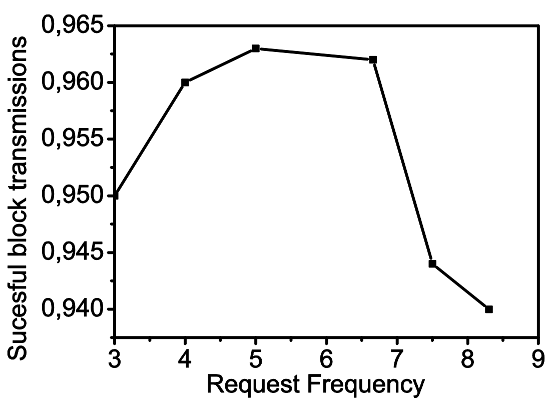


Figure 6: Scheduler performance with variable block request frequency

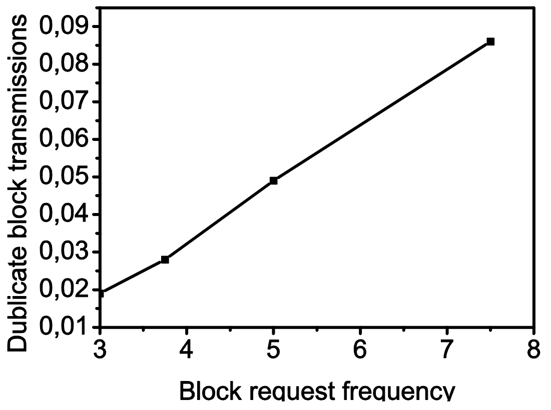


Figure 7: Duplicate block transmissions with variable block request frequency

In Figure 8 we demonstrate the percentage of successful block transmissions under variable numbers of neighbours in the overlay with the same parameters as in the previous scenario. In all the scenarios that we demonstrate the Ms and Mb are equal in order to be able to present the results in a two dimensional graph. We can see also plots under different values of Mi. As we observe there is again an area where Mi is between 4 and 8 and Mb,Ms are between 6 and 10 where the percentage of the successful block receptions is very high and remains almost stable. This reveals the stability of the system in dynamic peer arrival and departures where the values of these parameters change temporarily.

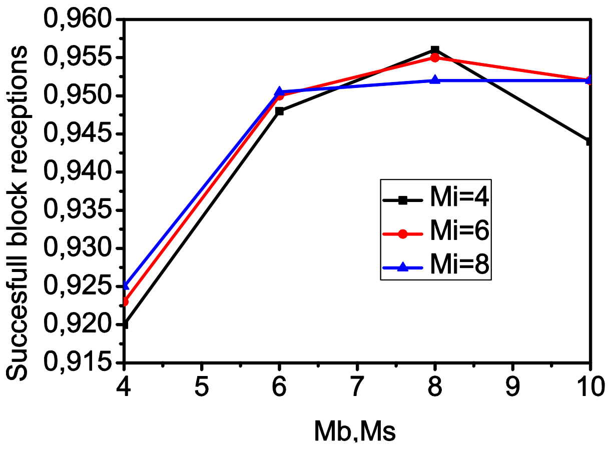


Figure 8: Variable number of neighbours in the overlay

In Figure 9, we evaluate the inter-ISP traffic that our system introduces to the underlying network. Towards this goal we assume an extreme scenario where 2000 peers that participate with equal probability in 40 ISPs. As we can observe from this graph the 90% of the whole overlay connections are between peers that participate in the same ISP. So, only 10% of the whole traffic crosses the inter-ISP network links.

We also evaluate our system under extreme dynamic conditions in terms of peer behaviour and underlying network changes. More specifically, 2000 nodes enter our system from 0 to 200 sec (10 peers/sec) while half of them depart during the period 150 to 250 sec (10 peers/sec). In addition, the uploading bandwidth of each peer dynamically fluctuates every 2 seconds between -20% and +20% of its nominal value according to a uniform distribution. We observe the degradation of our system performance is less than 3% compared to the static scenario. This shows the high tolerance of our system to dynamic conditions and also the very fast convergence of our optimization algorithms.

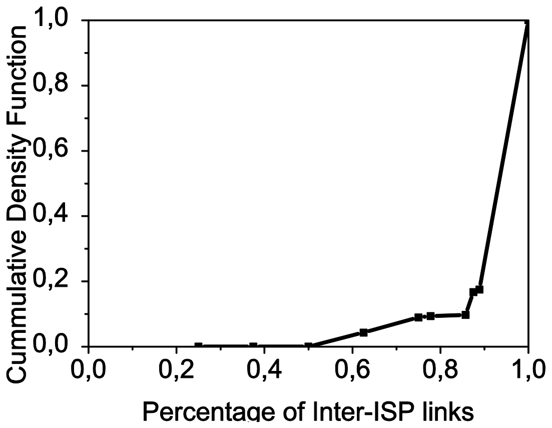


Figure 9: Percentage of InterISP overlay connections

In Table 1 we evaluate the impact that peer arrivals, peer departures and dynamic uploading bandwidths have on the performance of our system. For peer arrivals we simulate a system with 1000 nodes already present and we insert the remaining 1000 nodes with different arrival rates. Under these conditions, the reduction in mean block receptions has been kept small e.g. around 3% for 20 peer arrivals per second.

peer arrival per sec				**peer departure per sec**			
5	10	20	40	5	10	20	40
98%	97%	97%	95%	0,99%	99%	98%	98%
Bandwidth fluctuations							
5%	10%	20%	40%				
99%	99%	99%	99%				

Table 1: Impact on the performance of our system under dynamic conditions

For evaluating node departures we have used an overlay with 2000 peers that suffered from a departure of 1000 peers under various departure rates. We observe that our system is almost immune to peer departures due to the fast reconfiguration of our overlay and our dynamic neighbour selection function.

Finally, we present a scenario according to which every 2 seconds each node i with uploading bandwidth c(i) uniformly fluctuates its bandwidth between −h*c(i) and +h*c(i) where h assumes various percentile values. To the best of our knowledge we are not aware of any other work that exhibits such stability under such extreme dynamic conditions.

We have also carried out experiments where the average available uploading bandwidth is less than the video playback bit rate. In this case, our design still maintained high levels of fairness as all nodes have received a similar percentage of blocks. Finally, we have executed experiments in which the uploading capacities of the peers follow a uniform distribution, in contrast to the greatly unbalanced distribution that we used so far, and the behaviour of our system was identical. This implies that our architecture is independent of distribution of the uploading capacities of the participating peers.

2.2 Transcoding

2.2.1 Transcoding Purpose

The purpose of the static transcoder module developed by Telefónica I+D is to provide a system that allows multimedia content conversion, independently of content type like image, audio or video, but with different technical features. This process is called transcodification.

The objective of the transcodification process is to allow one piece of content to be played or displayed on various devices, by adapting the content to the player and none of the available resources for content consumption in the device are unprofitable. In those conversions we can find operations like format changes (container), audio or video bitrate modification, resolution downgrading for adapting to reduced screen, etc.

Depending on the use cases and scenario, the transcoder could take part not only in the content adaptation process, but also in taking the decisions about which parameter should be changed to get content adapted to the objective defined.

2.2.2 Multimedia Formats Supported

Multimedia types supported are: audio, video and image. Those are the basic content types needed for any multimedia based services. Each of those multimedia content types could have different formats. The transcoding module supports various formats as detailed in the following sections, depending on content type.

2.2.2.1 Video Content

In case of multimedia video contents, the transcoder module supports a specific list of containers and video codecs. In addition to that, video contents have another set of features that are very important in the transcoding process and have to be taken into consideration. For a better understanding of the complexity of the transcoding process, following parameters should be defined, including their characteristics and usual values.

- Resolution: number of different pixels that can be shown in each direction.
- Aspect Ratio: An image aspect ratio is its width divided by its height. Most usual aspect ratio values are 1.33:1, universal format for standard video and 16:9 (1.78:1), universal format for high definition television and European digital television.
- Frame rate: Frame rate shows the number of photograms per second that are needed for creating movement. It is given in photograms or frames per second (fps) or Hertz (Hz).

Video Containers

Following table shows video containers supported by the VITAL++ transcoding module, specifying if they are supported as input and/or output.

Name	**Input**	**Output**
MPEG-2 TS	X	X
MPEG-1 PS	X	X
AVI	X	X
3GP	X	X
ASF	X	X
MP4	X	X
FLV	X	X
WM	X	X

Table 2: Video Containers supported by VITAL++ Transcoding Module

Containers shown in the table are those commonly used. Following, a description for each of them can be found:

- MPEG-2 TS: MPEG-2 Transport stream; transport stream is a communication protocol designed for allowing multiplexation of digital video and audio in a synchronized way.
- MPEG-1 PS: MPEG-1 Program stream; program stream is a stream multiplexor of elementary streams in just one. The MPEG-1 development started in May, 1988.
- AVI: Audio Video Interleave; is a multimedia container format introduced by Microsoft in 1992. AVI files can have audio and/or video data in the same container, and AVI allows audio synchronization with video.
- 3GP: 3GP is a simplified version of encapsulated format MPEG-4 Part 14 (MP4), designed for reducing storage and bandwidth requirements, with the objective of being used in mobile devices.
- MP4: MPEG-4 Part14, formally ISO/IEC 14496-14:2003; is a format for multimedia encapsulations specified as part of MPEG-4.
- FLV: Flash Video is a proprietary file format used for video transmission in Internet, used by Adobe Flash Player.
- WM: Windows Media; encapsulating format for Advanced Systems Format (ASF). The .wmv extension is normally used for ASF files used by Windows Media Video codecs.

As shown above in the table, all of them are available as inputs and outputs of the transcoding engine.

Video Codecs

From a transcoding module point of view, video codecs can be supported as inputs or outputs for a specific video content. But in addition to the adaptation task, another important issue that has to be taken into consideration is the method used for delivering the transcoded content. The following table shows codec support but including also the delivery method.

Name	**Input**	**Output**	**File**	**Stream**
MPEG-1	X	X	X	X
MPEG-2	X	X	X	X
MPEG-4	X	X	X	X
H-264	X	X	X	X
H-263	X	X	X	X
WMV1/2	X	X	X	X
WMV 3	X		X	
VP6 (FLASH)	X	X	X	
VP7 (FLASH)	X			

Table 3: Video Codecs supported by Transcoding Module

Following, a short description of video codecs can be found, in order to have a better understanding of their features:

- MPEG-1: MPEG-1 is a standard for video and audio compression with losses. It was designed for VHS compression with CD audio until 1.5 Mbit/s (26:1 and 6:1 compression ratios respectively) without excessive quality, making possible video CDs, cable7satellite digital TV and digital audio broadcasting (DAB).
- MPEG-2: MPEG-2 is a standard for generic codification of images in movement and its related audio information. It describes a combination of compression methods with audio and video losses that allows the storage and transmission of films using current storage and bandwidth capacities. MPEG-2 is commonly used as digital TV signals that are transmitted by terrestrial, satellite and cable TV.
- MPEG-4: MPEG-4 is a group of audio and video standard codecs introduced at the end of 1998 by ISO/IEC MPEG (Moving Picture Experts Group) group. Main uses of MPEG-4 standards are audiovisuals media streams, CD distribution, bidirectional videocall and TV.
- H.264: is a standard for video compression also known as MPEG-4 Part 10 or MPEG-4 AVC (Advanced Video Codification). It defines new characteristics that allow new video compression in a much efficient way the older ones, giving more flexibility in a wide number of scenarios.
- H.263: H.263 is a video codec standard designed originally as a low-bitrate compression format for videoconferencing systems. Currently is used in several internet applications. .

- WMV1/2: compression video format developed by Microsoft for several proprietary codecs. The original codec known as WMV was designed for streaming application in Internet, like the RealVideo competitor ones, and built over a Microsoft implementation of MPEG-4 Part 2 standard.
- WMV3: WMV 9 introduce several important features including support for interlaced video, non-squared pixels and fram interpolation. WMV 9 also introduces a new profile named Windows Media Video 9 Professional, that is automatically activated when video resolution and bitrate exceed 300,000 pixels (p.e.. 512 × 586) and 1000 kbit/s. It is oriented to high definition video with 720p and 1080p resolutions.
- VP6: video codec developed by On2 as successor of other video codecs like VP3 and VP5. It has been used in broadcasting products and QuickLink software. VP6 is also used by Adobe Flash and Flash Video files.
- VP7: video codec developed as successor of VP3, VP5 and TrueMotion VP6. It supports VFW and DirectShow, acclaimed by On2 as better codecs tan competitors like MPEG-4 AVC (H.264) and VC-1.

2.2.2.2 Audio Contents

As already described for video contents, content information for audio has been split in two sections: container or encapsulators and codecs.

Audio Containers

Audio containers supported by the transcoding module are following:

Name	Input	Output
MPEG-2 TS	X	X
MPEG-1 PS	X	X
WAV	X	X
MP4	X	X

Table 4: Audio Containers supported by Transcoding Module

Some of the containers shown above have already been described in the video container section, as containers can be used for encapsulating not only video streams but also audio streams, jointly or separately. In this section only new containers will be described:

- WAV: WAV (or 'WAVE'), audio format type Waveform is a standard used by IBM and Microsoft for audio storage in PCs.
- MP4: is commonly used for digital audio storage and streams, especially for those defined by MPEG. But can be also used for storing other data like subtitles or static images as well.

Audio Codecs

The following table shows the list of audio codecs supported, specifying if they are available as inputs or outputs, indicating also the output method, file or stream.

Name	**Input**	**Output**	**File**	**Stream**
MP1A	X	X	X	X
MP2A	X	X	X	X
MP3	X	X	X	X
WMA	X		X	X
AAC	X	X	X	X
AC3	X	X	X	X
AMR	X	X	X	X

Table 5: Audio Codecs supported by Transcoding Module

Following are short descriptions of each codec:

- MP1A: MPEG-1 Audio Layer I, commonly named MP1, is one of the three codecs included in the MPEG-1 standard. Although it is supported by all major audio players, it is currently considered an older one and is more and more replaced by mp2 or mp3.
- MP2A: is a codec defined by ISO/IEC 11172-3. Whereas MP3 is a popular format for PC and Internet based applications, MP2 is kept as dominant codec for audio broadcasting.
- MP3: MPEG-1 Audio Layer 3, commonly known as mp3, is a digital audio codec using a compression system with losses.
- WMA: Windows Media Audio (WMA) is an audio compression technology developed by Microsoft. Its name can be used for referencing audio files extension or the audio codec itself. It is composed of four different codecs: the original codec, known as WMA; WMA Pro, a new and advanced codec that supports multichannel and high resolution; a codec without losses, WMA Lossless, that compressed audio data without fidelity losses; and finally WMA Voice, that uses compression using a low bitrate range.
- AAC: Advanced Audio Coding (AAC) is a compression and encoding schema with losses standardized for digital audio. Designed as mp3 successor, AAC is used to get better sound quality than mp3 in most bitrates.
- AC3: Dolby Digital, or AC-3, is the most common version that has up to 6 discrete audio channels. AC-3 supports audio rates up to 48 kHz.
- AMR: Adaptive Multi-Rate (AMR) is an audio compression schema optimized for speech encoding. It has been adopted as codec standard by 3GPP in October 1998, and currently is used in mobile communication technologies as GSM and UMTS.

2.2.3 Transcoding Engine Architecture

This chapter will analyze the transcoding engine, dividing the module into its components, and conducting a detailed description of its functionality and features.

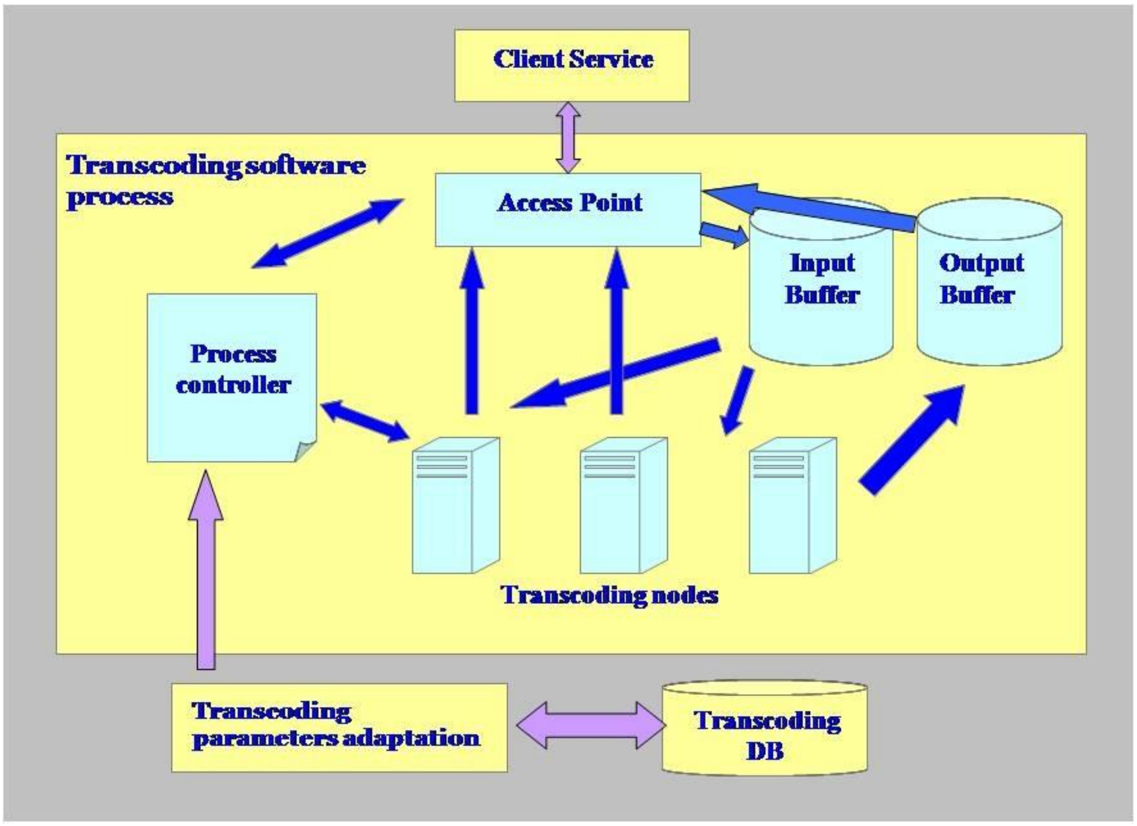


Figure 10: Transcoding software process Subarchitecture

Transcoding software process Subarchitectureshows different elements implied in the transcoding process. Firstly, multimedia content is ingested in the system. It could be an image, video or audio file. Once the content is in the system, two processes start:

- **Content Analysis Module:** This element analyzes automatically multimedia features of the ingested content. Some of those features are: type of content (image, audio, and video), container format, codec, bitrate, resolution, etc.
- **Output Parameters Selection:** This module selects, from a parameter list, the features that the content will have in the output file(s). These parameters can be selected by means of a graphical interface, XML interface or pre-sets. Not all parameters have to be indicated, just the container must be given, as it specifies the file extension.

These two processes are connected with the transcoder database. With the inputs from both elements, the database gets necessary transcoding parameters. With this information the transcoding software begins the adaptation to get appropriate output to get the multimedia content adapted as requested.

In the upcoming chapters a detailed description of different modules implied in the transcoding process is shown.

2.2.3.1 Content Analysis Module

The input to this module is the original multimedia content. The content is analyzed by this module which determines a set of multimedia parameters that characterize this content, like:

- Type of content and format
- File Size
- Duration
- Bitrate
- Aspect Ratio
- Video Codec
- Audio Codec
- Image Codec
- Resolution
- Frame rate
- Audio Channels
- Sample Frequency

All these parameters compound the output to this module and are at the same time the input for next module in the transcoding process, the database.

2.2.3.2 Output Parameters Selection

This module is an interface for selecting the desired output parameters. There are a wide variety of parameters that can be selected or configured, but not all of them are needed. The only parameter that is mandatory during the whole transcoding process is the container, because it is needed for knowing the extension and format for the output content.

Most of those parameters match with those taken from the original content by the "Content Analysis Module". But in addition to the individual selection of parameters, it is also possible to use profiles.

Transcoding profiles are a set of output parameters predefined and adapted to devices' features where contents will be played. These profiles can be defined for different types of devices, in a way that users do not need to provide any information related to their devices, but they could get contents adapted to them in the best way.

2.2.3.3 Transcoder Database

Transcoder database is based on the MySQL database engine. All the information related to the transcoding process is stored in the database, including multimedia parameters described in sections above.

When a new codec is included in the transcoder, it has to be registered in the database, including all parameters needed for its appropriate configuration. With this information, in addition to transcoding profiles and information provided by the content analysis module, the content generation model is continuously updated.

2.2.3.4 Transconding Parameter Adaptation

Once parameters for output file have been defined, the next step is the adaptation of transcoding parameters. This adaptation uses information from the database, where possible combinations for parameters and configuration are registered.

Not all transcoding processes or combinations for input/output are possible, because not all formats are compatible. For example, some containers have specific restrictions, like video codec or resolution, and something similar happens in audio codecs with sample frequency or channels limitation. All these limitations are very important and have to be taken into consideration for getting a correct transcoding process. Due to this, although some parameters are selected at the beginning, they could be modified for complying with output requirements.

This issue can be solved using profiles, where parameters have been defined having previously mentioned restrictions in mind.

2.2.3.5 Transconding Process Software

This is the main module of the transcoding engine on which the transcoding process takes place based on the parameters obtained from other system modules.

The transcoding process software is responsible for adapting the input content to the requested format or specifications. The decision of output characteristics to be used is given by the input generated by the Content Adaptation Module.

In addition to the information received as input, the transcoding module needs also the original content itself, including a description. Depending on the adjustment to be performed, this module will need to call different modules for adaptation.

The output of this module will be the input file, but adapted according to requirements and parameters established previously.

A further analysis of this module can be found in the following chapter.

Architecture

The software transcoding module is based on a multiple server architecture with the ability to transcode audiovisual contents to many media output formats (audio, video and image).

Transcoding software process Subarchitectureabove shows different components of this module and the interaction between them.

Although just one client service is accessing the system, normally the transcoding system will require resources for supporting many services.

The access point is the system’s interface to the external requests. From this point, the input contents will be sent to the Input Buffer, and the output contents (those contents already transcoding and adapted to the requirements) will be transferred to the access point from the Output Buffer.

Following with the transcoding engine workflow, and after the adaptation of the transcoding parameters starts, the transcoding software process begins. This process starts when the transcoding request arrives at the Process controller. This module manages the process distribution among the nodes, based on the specific coding module needed in each case.

The transcoding engine is based on a multi-node architecture, based on input and output buffers. This architecture facilitates requests concurrency, maintaining a minimum waiting time guaranteed and improving system performance.

Each of these nodes will manage a set of transcoding threads: characteristics and capabilities of these threads are identified and described in the transcoding database. The controller is responsible for allocating tasks in the free transcoding nodes, launching remote transcoding applications.

Communication between the Process controller and the Transcoding nodes is based on XML messages. Once the process controller decides which of the nodes will process the request, an XML message is sent to it, specifying the parameters of transcoding required. This node receives the message and after a short processing gets the corresponding content of the input buffer and performs the transcoding. When the node has just finished the process, the resulting output content is sent out to the output buffer and an XML message is sent to the process controller to report the successful completion of the task.

2.2.4 Transcoding Module Interface

The service interface is based on a web service using SOAP messages sent through HTTP.

The media input to the system can be done using either bitstreams or files. When files are involved, the protocols of choice for uploading content are either ftp or sftp for the transport. It is also possible to have the content available through an external file system that can be mounted from the Transcoding nodes. If the contents are streams the protocol that best suits is UDP unicast.

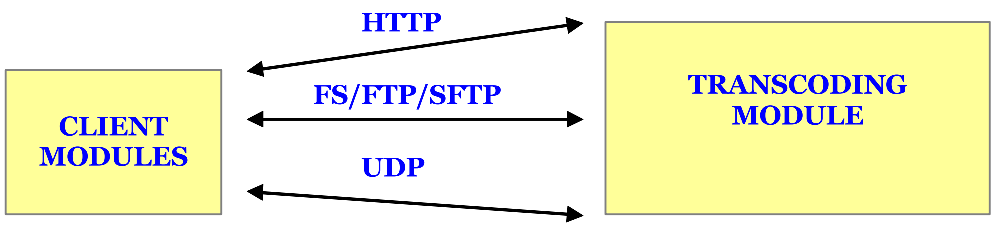


Figure 11: Protocols used in the engine interface

The system also allows asynchronous transcoding jobs if needed: the requester transcoding end has means to notify the reply address used for the cases where the petitions should be queued because there are no available resources for that request, and also to receive the transcoding finalization and cancellation replies. If this reply address is not provided (url HTTP) and there are no available resources to initiate the transcoding, the request will be denied and it will not be queued.

2.2.4.1 Transactions

The types of the SOAP operations that could be exchanged through the service interface are the following ones:

1. Transcoding request.
2. Transcoding session state request.
3. Cancellation of transcoding session.
4. Possible transcoding profiles request.
5. Possible transcoding bitrates.

Transcoding Request Operation

With this transaction the client part requests a transcoding operation indicating the media and format for the content, for both original and target content (adapted one). It is also possible that media and format for adapted content was indicated by means of transcoding profiles.

The media content source and target can be either files or unicast streams, but if the source is a stream, the destination can only be another stream.

Request Parameters for getting the original content

The parameters for defining the media where the original content will be transferred are:

- Transfer mode: that can be a file or a stream.
- "imode". This parameter can have two possible values: "file" or "stream". If value is "stream", the transcoding requester doesn't need to indicate any additional information in the request. The transcoding engine will indicate in the answer the address and port where the stream has to be sent.
- Original content description, in mode "file":
  - "ifprot". Transfer protocol, like "localfile", "sftp" or "ftp".
  - "ifhost". IP address or server name where original content is stored.
  - "ifport". Port for accessing to transfer service in the source server.
  - "ifuser". Username for accessing the content.
  - "ifpwd". Password for accessing the content.
  - "ifpath". Complete path where original content is located in client server.

All these parameters can also be given in a URL format.

Request parameter for indicating original content format

All these parameters are optional, due to the fact that the transcoding module can analyze the content source and get information about the format. Anyway, it can be used on the client side if needed.

- For video content, following parameters can be used:
  - "icont". Content container, possible values depend on transcoding configuration, but most usual will be "mpeg2ts", "mp4" and "avi".
  - "ivcodec". Content codec format, such as "mpeg1", "mpeg2", "mpeg4v", etc.
  - "ivfr". Frequency.
  - "ivinterlaced". For indicating interlaced video. Possible values are "1" for interlaced and "0" for not interlaced.
  - "ivwidth". Horizontal resolution in pixels.
  - "ivheigth". Vertical resolution in pixels.
  - "ivar". Image aspect ratio.
- For audio content, next parameters can be used:
  - "iacodec". Audio codec, like "mp3", "mpeg2a", "mpeg4a", etc.
  - "iachann". Number of audio channels.
  - "iasamps". Sample size in bits. Usual values are "16" or "8".
  - "iasf". Sampling frequency.

Parameters for defining the adapted content format

The format for the transcoded content output can be indicated by the profile, or indicating the value of each parameter in the transcoding request.

"oprofile". Transcoding profile requested. In case that transcoding profiles have been defined, the requester can select one of them.

- Optional parameters for adapted video content are:
    - "ovfr": Frequency.
    - "ovinterlaced". For indicating if video is interlaced or not. Possible values are "1" for interlaced and "0" for not interlaced.
    - "ovwidth". Horizontal resolution in pixels.
    - "ovheigth". Vertical resolution in pixels.
    - "ovar". Aspect ratio.
    - "ovminbitrate". Minimum value for bitrate.
    - "ovavgbitrate". Average value for bitrate.
    - "ovmaxbitrate". Maximum value for bitrate.

- Optional parameters for audio content are:
    - "oachann". Number of channels.
    - "oasamps". Sample size in bits. Common values are "16" or "8".
    - "oasf". Sampling frequency.
    - "oaminbitrate". Minimum value for bitrate.
    - "oaavgbitrate". Average value for bitrate.
    - "oamaxbitrate". Maximum value for bitrate.

Parameters for defining the adapted content location

The Transcoding module can work in two ways regarding output content location. The client part can indicate where to send the output content as part of the transcoding request, or the transcoding module can indicate where the output content will be located as part of the answer to a transcoding request.

The parameters used for indicating the way the transcoding module will send adapted contents depends on the transfer mode. These parameters are optional:

- When transfer mode is "file" the attributes to be used are:
    - "ofprot". Transfer protocol, like "localfile", "sftp" or "ftp".
    - "ofhost". IP address or server name where adapted content is stored.
    - "ofport". Port for accessing to transfer service in the server.
    - "ofuser". Username for accessing the content.
    - "ofpwd". Password for accessing the content.
    - "ofpath". Complete path where original content is located in client server.
- Destination address in case transfer mode will be stream:
    - "osIP". IP Address of destination server where the transcoded stream has to be sent.
    - "osPort". Port of the destination server where the transcoded stream has to be sent.

Answer to Transcoding request

As answer to a transcoding request, the transcoding module will return the following information to the client part:

- "result". Indicating the status of the request. Possible values are:
  - "TRANSCODING". Request accepted.
  - "REJECTED". Request rejected.
- "sessionID". Identifier for transcoding session related to the request done. In case the result of the request will be "REJECTED", sessionId will not have a value.
- "isIP". This is the IP address where the client has to send the stream with the original content in case stream mode is used.
- "isPort". It is the IP address where the client has to send the stream with the original content in case stream mode is used.
- "cause". Code indicating the cause for request rejected.
- "oUrl". Url where client part could get the adapted content.

Request for Transcoding Status

With this operation, the client requests information about the status of a transcoding request.

The client must indicate the "sessionID" received as answer to the transcoding request.

It will get as answer following attributes:

- "state". Session status, with possible values:
  - "QUEUED".
  - "TRANSCODING".
  - "FINALIZED"
  - "UNKNOWN". Indicates that sessionId parameter does not exist.

Request for transcoding session cancellation

With this operation the client part cancels a transcoding process already requested.

For that, the cancel resquest should include the sessionId related to the transcoding process that should be cancelled.

Request for transcoding profiles available

With this operation the client can request a list of available profiles that can be used for transcoding a determined content, depending on the original format.

The parameters that can be used in this request are:

- "*imode*". Transfer mode for the original content. Two modes are available, "file" or "stream".
- "*icont*". Container format for the original content. It could be "mpeg2ts", "mp4" and "avi".
- "*ivcodec*". Video codec for original content format, like "mpeg1", "mpeg2", "mpeg4v", etc.
- "*iacodec*". Audio codec for original content, like "mp3", "mpeg2a", "mpeg4a", etc.

The answer to this request is a list of profiles that can be chosen for transcoding content, based on the format indicated in the request. Each profile includes an implicit combination for mode, container, video and audio codec. In an informative way, detailed information about each profile is also included.

- "*oprofile*". Profile name.
- "*omode*". Transfer mode for output mode. It can have two values: "file" or "stream".
- "*ocont*". Container format.
- "*ovcodec*". Video codec.
- "*oacodec*". Audio codec.

Request for bitrates supported by a transcoding profile

With this operation a client can request a list of bitrates supported by a transcoding profile that is to be used in a transcoding request.

Following a list with parameters to be used:

- *"oprofile".* Profile name.
- *"omode".* Transfer mode for output content. It can have two values: "file" or "stream".
- *"ocont".* Container format.
- *"ovcodec".* Video codec format.
- *"oacodec".* Audio video codec.

Information included in the answer to this request is:

- *"oprofile*". Profile name.
- *"oabit".* Bitrate supported by requestd profile.
- "*ofreq*". Frequency suported by both bitrate and profile.

2.3 Conditional Access Management

The Content Protection Subsystem contains a conditional access system that evaluates licensing requests against business rules set by the content provider.

These rules can be set across multiple parameters including:

- User Identity (user-based access);
- Group Membership of the User (group-based access);
- Time (i.e. content may only be licensed for a certain timeframe);
- Device capabilities and media encoding;
- Accounting rules such as pre-paid versus post-paid billing;
- Network Provider and/or location information.

Identity based conditional access enables Vital++ to leverage the strong authentication, and potentially message encryption, of IMS in identifying whether a network subscriber is entitled to access content. For example, a Vodafone Germany subscriber could be allowed to access RBB content anywhere in the world providing they authenticate using their IMS.

2.3.1 Conditional Access Business Rules

The format and processing mechanism for business rules are described in D4.3. However, for the purpose of describing the SoftMix application we will reiterate relevant business rules below in a pseudo-code format similar to how they are described using Drools.

Rule Type	Condition	Result
Content Provider Category	Category=='professional'	Allow publishing to super-peer
	Category=='prosumer'	Don't allow publishing to super-peer
Subscriber Account Type	Account_type=='prepay'	Account balance must be in credit AND Account(balance) >= Content(cost) Can't access content where Content(Account_type)=='postpay'
	Account_type=='postpay'	Can access Content where Content (Account_type)=='postpay'
Subscriber Account Status	Status=='active'	Can access OR publish content, depending on evaluation of other rules
	Status=='inactive'	Can't access OR publish content
Date/Time	Date()<= Content(publish_date)+ time_restriction	Content can be licensed so long as 'time restriction' days/hours/mins hasn't elapsed. E.g. 7 day rule described in D4.3
	Date()>= Content(publish_date)+ time_restriction	Content can't be licensed.
Location	Subscriber (location) != ContentProvider(location) AND forall (content(blacklist)!=location)	Some content cannot be licensed from blacklisted locations.
Network	ContentProvider(whitelist) contains network provider(id)	Content can be licensed to a subscriber accessing through 'network provider' without considering their location. However other rules should still be considered.
	!(ContentProvider(whitelist) contains NetworkProvider(id))	Check location rules. If Subscriber(location_!= ContentProvider(location) then content can't be licensed.

Table 6: Conditional Access Business Rules

2.3.2 Accounting Rules

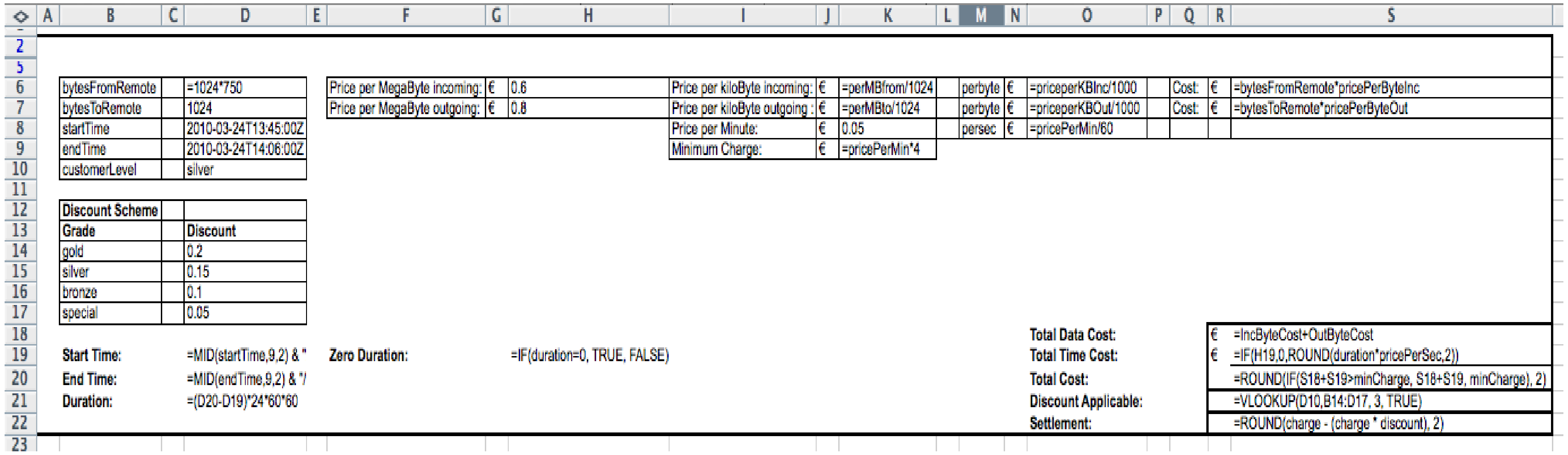
The Accounting subsystem is described in more detail in D4.1. However, for SoftMix the following charging rules have been defined.

Accounting Rule Type	**Condition**	**Result**
Charge by Content	Content(licenses) contains 'Fair Use'	Free, if Fair Use is asserted
	Content(Charging_model)=='FREE'	FREE
	Content(Charging_model)=='LIMITED') AND ContentProvider(freelist) contains Subscriber(group)	FREE if on list, Charging_model='PURCHASE' otherwise
	Content(charging_model) == 'PURCHASE'	Calculate Content Charge
Charge by Location/Network	ContentProvider(location_specific.location) contains Subscriber(location)	Use location_specific tariff
Discount Group	ContentProvider(discount_groups) contains Subscriber(group)	Apply a discounted tariff
Content Encoding	ContentProvider(encoding_specific.encoding) contains Content(encoding)	Apply an encoding specific tariff
QoS	ContentProvider(qos_specific.qos) contains Subscriber(qos)	Apply a qos specific tariff

Table 7: Accounting Rules

A sample rating "work sheet" is shown below. It was created using Microsoft Excel. The CREST accounting system described in D4.1 enables content providers to describe their tariffing options in terms of Open Office XML[2] and the competing Openoffice XML File format worksheets. Data and formulae within the worksheets are then associated with charging data from various sources e.g. diameter content charging data, overlay management statistics and per-session based network traffic data from an IMS Charging Gateway Function.

[2] OpenXML- the new office format ratified by the ECMA, http://www.openxml.biz/



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
2																			
5																			
6		bytesFromRemote		=1024*750		Price per MegaByte incoming:	€	0.6	Price per kiloByte incoming:	€	=perMBfrom/1024		perbyte	€	=priceperKBInc/1000		Cost:	€	=bytesFromRemote*pricePerByteInc
7		bytesToRemote		1024		Price per MegaByte outgoing:	€	0.8	Price per kiloByte outgoing :	€	=perMBto/1024		perbyte	€	=priceperKBOut/1000		Cost:	€	=bytesToRemote*pricePerByteOut
8		startTime		2010-03-24T13:45:00Z					Price per Minute:	€	0.05		persec	€	=pricePerMin/60				
9		endTime		2010-03-24T14:06:00Z					Minimum Charge:	€	=pricePerMin*4								
10		customerLevel		silver															
11																			
12		**Discount Scheme**																	
13		**Grade**		**Discount**															
14		gold		0.2															
15		silver		0.15															
16		bronze		0.1															
17		special		0.05															
18															**Total Data Cost:**			€	=IncByteCost+OutByteCost
19		**Start Time:**		=MID(startTime,9,2) & "		**Zero Duration:**		=IF(duration=0, TRUE, FALSE)							**Total Time Cost:**			€	=IF(H19,0,ROUND(duration*pricePerSec,2))
20		**End Time:**		=MID(endTime,9,2) & "/											**Total Cost:**				=ROUND(IF(S18+S19>minCharge, S18+S19, minCharge), 2)
21		**Duration:**		=(D20-D19)*24*60*60											**Discount Applicable:**				=VLOOKUP(D10,B14:D17, 3, TRUE)
22															**Settlement:**				=ROUND(charge - (charge * discount), 2)
23																			

Table 8: Sample Work Sheet "Accounting Rules"

This scheme uses the following data to calculate a charge for content

- Minimum charge (a basic content charge)
- Charge/second for the duration of the session
- Charge/byte offset by charge/byte uploaded to other overlays
- Membership Level (Gold, Silver, Bronze)

3 Content Services

In contrast to Content-related Services which enable and support content and service provision technically, Content Services are applications which actually communicate with the user. The following chapter outlines a number of potential business scenarios which have been discussed by the VITAL++ consortium and – technically – could be realised using the VITAL++ architecture and components.

While the first part (#3.1) outlines potential business cases which will not be realised in the course of this project, the second (#3.2) details the objectives and processes of the project's demonstration service, the personalised radio service SoftMix, including a detailed descriptions how the generic components described in chapter 2 work together in order to realise the SoftMix Service.

3.1 Potential Business scenarios for VITAL++

The following describes a series of potential business scenarios and use cases in order to underline the exploitation potential of the VITAL++ schemes and architectures.

3.1.1 RBB Remote

Due to licensing policies AV Content on the internet is often geo-blocked, i.e. only available in certain areas. This, however, excludes users who have paid their broadcast licence fees but happen to be temporarily outside the geographic area where they live and pay their fees.

With IMS technology, viewers can be enabled to consume content they have a right to access wherever they are. A suitable area of application would be the streaming AV (IPTV) offers by national public broadcasters, which could then be made available for all rightful viewers anywhere throughout Europe.

Although this concept stands as a valid business case, its realisation was not possible in the course of VITAL++. Whereas all technological issues concerned will have been realised by the end of the project time, the realisation of such a service would evoke a major administrative effort which content partner RBB is not legally entitled to perform. Technically, it would be easy to allow fee payers to access such content. However, from an administrative point of view this would mean to allow the system access to the fee payers' database which carries their related IDs and, what's more, it would mean that fee payers would have to know or carry their IDs so that they can log in.

3.1.2 Content Distribution in Rural Areas

In some remote rural areas, served by satellite connections or radio access, the use of P2P technologies can improve the way operators serve multimedia on-demand content. In a rural area where a number of users are connected to a broadband network using a number of satellite accesses the same content may be forwarded at different times at several satellite accesses using a high amount of bandwidth. This scenario can be improved if subscribers are connected to a local area network (wired, WiFi, etc.) and share one satellite access.

The network operator can improve the use of the expensive and scarce bandwidth satellite access using a P2P approach. This approach can be a user P2P, where a user serves contents to other users or even an operator P2P, where every on-demand content requested by a user to the network is stored at a local element belonging to the operator. In both cases, when a second remote user asks for the same content, it is distributed from the local broadband network, instead of using the satellite access time and again.

This content distribution scheme, as proposed by TID in spring 2009, could improve content services in rural areas decisively. Unfortunately, it was not possible to realise such a case in the course of the project.

3.1.3 Science Video Blog

The architecture and functionality of VITAL++ enables the creation of a video pool for science communication. Individual institutions could connect with others – whether inside a closed group like Fraunhofer Gesellschaft or between free, individual organisations sharing interest in a certain topic – and share information on the latest developments and findings.

Such an application would not require much of a fancy GUI design but rather should be based on decent metadata handling to ensure that interested community users will find what they are interesting in.

3.2 Soft Mix Service

The SoftMix service, developed by rbb in the course of VITAL++, serves as main demonstrator for the full functionality of the VITAL++ individual components and overall architecture.

Independent of the technological setting, its basic aim is to enable users to experience a truly personalised radio which serves exactly the kind of content that they like. This service is meant to be a prototype of a radio of tomorrow, combining the traditional experience of the radio with the possibilities of the Internet: while traditional radio runs in the background and does not require any kind of interaction with the "output device/user interface" (i.e. the radio), the Internet offers a variety of opportunities to interact with the content through the service, including browsing, skipping, downloading, etc. SoftMix aims to enable the user to listen to the radio and personalise this experience according to his/her preferences. In this case "user" is the most suitable term as the service enables active use and influence, way beyond passive listening. As long as the user is happy with the programme, however, s/he will not have to interact at all. The radio will just keep playing without further intervention.

The SoftMix service is intended for use on different IP-based devices, stationary or mobile. Depending on their peer-to-peer capabilities, bandwidth, screen displays and other potentially limiting factors some such devices may not offer all of the service features. However, it is clearly intended that all of them offer the basic capabilities of receiving multimedia content according to their preferences and to further influence their profiles.

3.2.1 First Steps

The SoftMix service will be offered via a dedicated rbb website where interested users can download the client and choose a start profile. Users will have to register for the service, saving some basic information as a first step towards their personal radio style. As rbb's six radio channels are already targeting certain interest groups they will be a good starting point. Therefore, the first step towards a user profile is to choose the one rbb radio channel profile that most suits their interest and taste. From the moment that they selected one they can start receiving their radio programme and will receive more general recommendations soon according to their first reactions (*like/dislike*) to the programmes they received. The respective procedures will be described and explained in the following chapters.

3.2.2 GUI

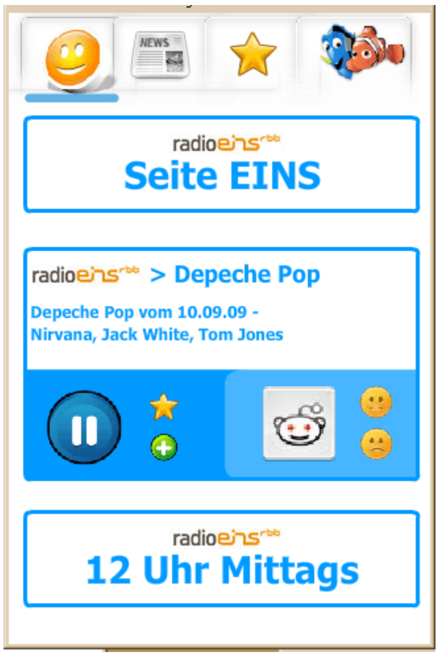
The Service will be presented in four different views that represent groups of related functionalities. Users can switch between the four views to use different options.

This separation in four separate views sprang from the idea to have one common UI for different types of devices. This approach requires keeping the UI so simple that it can be used on PCs as well as mobile devices which don't offer pointer devices and/or clicking options.

Each of the four views will offer simple options to switch to any of the others, either by using a joystick/navigation field on mobile phones or click/touch for PC or touch screen devices.

The following chapters give a general overview of the use and navigation of tabs.

3.2.2.1 Player View


NEWS
radioeins rbb
Seite EINS
radioeins rbb > Depeche Pop
Depeche Pop vom 10.09.09 - Nirvana, Jack White, Tom Jones
radioeins rbb
12 Uhr Mittags

- The player view will **display** the current media file on a **SCREEN**, plus the basic information (channel, series, title/short desription) – most other information will be available for searching and profiling but not visible in the player view.
- Above the display screen there will be an **OVERLAY** with 6 **BUTTONS** offering the basic functions (PLAY/PAUSE, BOOKMARK, LIKE/DISLIKE)

Above the main screen the main information (title, author/artist, duration) of the **previous** media item(s) will be accessible. Moving the focus there, will move this item to the screen – while the current audio keeps playing, unless the user hits play/pause! Having moved this item to the player screen, the user can use the Buttons on this file: this file then can be bookmarked, banned, recommended, etc., even though the current audio keeps playing.

Below the player screen the basic information of the media item coming up **next** on the playlist will be visible. Moving the focus down there will stop the current item and start this one - This will **replace** the **SKIP** button!

In case the service offers a video file the player will take the full screen and offer the Basic Buttons. Moving up and down in the playlist, however, will stop the video.

The Basic Buttons

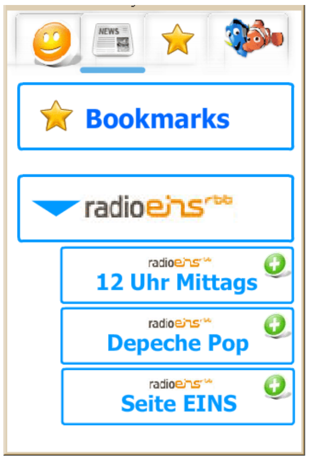
- **PLAY/PAUSE**
- **BOOKMARK**: (**this file**), so you can listen to it later
- **SUBSCRIBE**: (to **this series**); it will be added to your profile and any new episode added to your playlist with every new update
- **RECOMMEND**: will show a bar with icons of friends that are online and can be selected to receive a recommendation. This file will then appear in their playlists
- **LIKE**: this file and/or its related series will be uprated in your profile
- **DISLIKE**: this file/series will be downrated in your profile
- (**SKIP**): not an actual button but an available function/option

3.2.2.2 Bookmarks View

In this view users can manage their profile playlists in different ways:

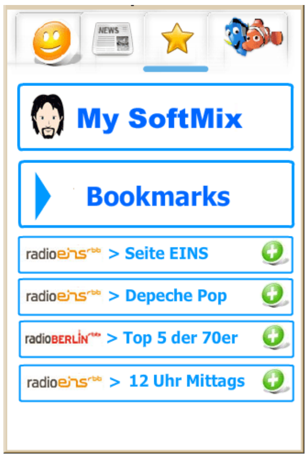
Users can add or delete subscriptions, and thus influence their profiles explicitly, or manage bookmarks, as explained below:

Subscriptions



- Programme subscriptions can be viewed added and/or deleted by clicking on an icon ('+'/'-') in the Category/Programme Search View, see below

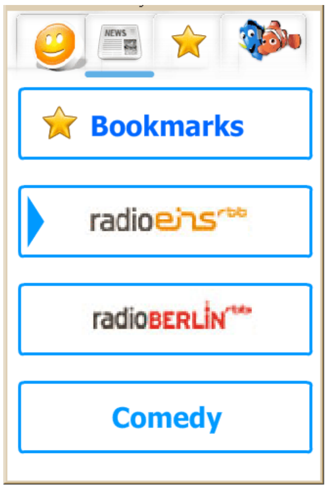
Bookmark

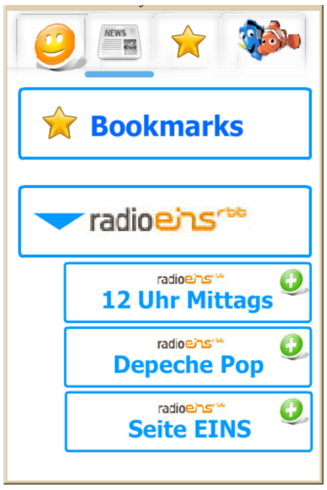


My SoftMix
Bookmarks
radioeins rbb > Seite EINS
radioeins rbb > Depeche Pop
radioBERLIN > Top 5 der 70er
radioeins rbb > 12 Uhr Mittags

- Bookmarked objects can be replayed, forwarded (recommended), etc.
- Bookmarks can be deleted

3.2.2.3 Category / Search View

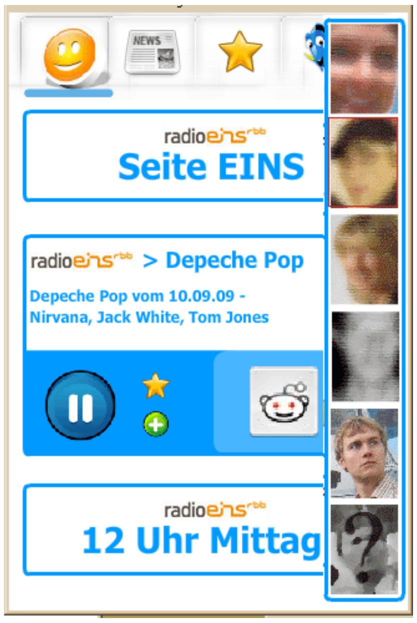




- Contents can be browsed by radio or thematic channel
- Clicking on a channel will show a list of available programmes associated with this category. Here, these can be added to or removed from the subscription list (see above)

In the background, the application will also perform keyword search, according to keywords that were associated with programmes and episodes the user *liked* or *disliked*. The user, however, will not be able to search the system for particular files, unless s/he has bookmarked them before.

3.2.2.4 Community Feature/ Broadcast Mode


NEWS
radioeins rbb
Seite EINS
radioeins rbb > Depeche Pop
Depeche Pop vom 10.09.09 - Nirvana, Jack White, Tom Jones
radioeins rbb
12 Uhr Mittag

- By clicking on the Social App icon users can communicate with buddies
  - A list of buddies who are online will appear
  - A click on any of the profile images will lead to their site, either showing what they currently listen to or what they currently "share" with others
- Using *Broadcast Mode*, users can "publish" simple media files or complete playlists as their own little radio show
- Other users can tune in to their show and listen to this prepared playlist rather than to their own playlist as generated by the SoftMix software
- By using the *like/dislike* buttons, etc., content and respective information (metadata) from these playlists can be used to improve the user's profile
- Publishing functionality will be provided by the Vital++ clients

3.2.3 Profiling and Personalisation

The Recommendation Engine, a central component for realising the SoftMix application, will match the user's Attention Profile (see 3.2.4.1) with content available for use for the SoftMix Service, will trigger a search over the connected network(s) and generate playlists and programme schedules out of the search results.

The user will utilise the application's buttons (see above) to improve his/her profile; e.g. click/activate "*like*" or "*dislike*", subscribe to certain programmes or recommend them to others. All these activities will influence the user's profile more or less directly (see 3.2.4.3)

3.2.3.1 APML – Attention Profiling Markup Language

The SoftMix Profiling Engine is based on the Attention Profiling Mark-up Language, APML.[3]

"..Attention profiles are consolidated, structured descriptions of people's interests and dislikes. The information about your interests and how much each means to you (ranking) is stored in a way so that computers and web-based services can easily read it, interpret it, process it and pass it on should you request and permit them to do so."[4]

APML is an xml-based format for storing users' attention profiles, i.e. information on what topics and sources a user is interested in and to what extent. The APML workgroup is an open group of interested developers, so the profiling language and its specifications are free to everyone and can be adapted to particular needs. For SoftMix the original specifications were left as they are common to everyone so that profiling data can be imported from or exported to other applications using this mark-up language. There are only one or two adaptations

[3] For further information see http://www.apml.areyoupayingattention.com/endusers/overview/.

[4] Quotation from http://cleverclogs.org/2007/10/basics-of-atten.html

specific to SoftMix, e.g. the source locator which helps other VITAL++ components to identify the relevant media files and at the same time carries content information about it (see more detailed description below).

3.2.3.2 The Frame Scheme

To organise content search only according to user preferences would generate a random radio programme which would most certainly not have the quality of traditional radio programmes. In order to avoid arbitrary playlists the concept of Programme Frames was introduced. Over long periods radio experts have developed concepts to organise radio programmes so that they entertain and inform people in different ways at different times of the day; the famous morning radio shows are a prominent example for this.

SoftMix now also offers such Programme Frames in order to organise the radio programme in the way that generic frames define what type of content should follow the current item. This would avoid that three recipe podcasts or weather forecasts would be played in a row.

The Recommendation Engine which matches the user profile with the available content will now do this according to the currently relevant frame and thus filter the search request and at the same time organise the order of the playlist.

3.2.3.3 The SoftMix Locator

SoftMix uses a specific locator for specifying each and every content item in terms of source and category. This locator is made up of the following parameters:

[content provider_bouquet]:[content provider_channel]:

[main category]:[subcategory]:[title of programme]:[number or name of episode]

This could look as follows:

rbb:fritz:youth:society:high-noon:*

This basic set of metadata information will enable the recommendation engine to search and find relevant pieces to put together a playlist that matches the user's profile. This basic set of parameters will furthermore be supported by other metadata available for this content item.

3.2.3.4 Basic Set of Rules

Users can influence their profile preferences by using certain buttons (see above). The following gives an insight into the mathematical calculation of the extent to which a content item matches the user's interests and preferences.

Whether a user dis-/likes a file is measured according to the following scale and calculated according to the following parameters.

3.2.3.5 Scale

Hates	Probably dislikes		Might dislike		Might like	Should like		Probably likes		Loves
-1	– -0.9	– -0.7	– -0.5	– -0.3	0	– 0.3	– 0.5	– 0.7	– 0.9	1
🟥	🟧	🟧	🟧	🟧	🟨	🟩	🟩	🟩	🟩	🟩

All files enter the system at a default rank of ZERO which is then adapted according to the user's preferences, i.e. every time a user clicks, e.g., dislike while a certain item is being replayed this influences the rating of this item according to parameters described in the following subchapters.

3.2.3.6 Parameters

Actions of a user influence the file that "provoked" the action (explicitly) as well as other files (implicitly) which may

- belong to the same series, category, radio channel
- feature the same author, artist or moderator
- feature the same topic

Explicit Influences

Action	**MINUS/PLUS**	**Consequence**
BAN/"HATE"	Downrate to 1-	Will never be played again
LOVE	Uprate to 1	May be played again, when there are no more unknown/unplayed items available
Skip	Minus 0.1	May be played again later – maybe the user hasn't consumed the file in total?
Subscribe	Jump to 1	All shows of this series will automatically jump to 100% and will not be subject to "indirect influences" (see below) until the show is unsubscribed
Unsubscribe	Jump to -0.2	All shows of this series are subject to the "indirect influences" (see below) again. It seems to make sense to rank them lower than shows of a new, un-evaluated series
Recommend	Plus 0.1/0.2	If a user bothers to recommend a file or show, s/he will most probably like it: recommending it once can mean that they just think it suits their friend's taste, recommending it two or more times probably means they actually like it

Table 9: Explicit Influences of User Activities on User Profile

- another influence: when a file was played, it should be "blocked" for a while (or in the case of spoken shows it should be blocked completely), so as to ensure that shows are not played twice (within a short time frame)

Implicit Influences

If a show was banned ("hate") or loved, this influences other shows

Influenced shows	**MINUS/PLUS**	
Shows of the same series	0.1	
Shows with the same topic (content keyword, NOT category!)	0.1	Related to individual content items, not series

Table 10: Implicit Influences of User Activities on User Profile

3.2.4 Service-related processes

This part aims to give an overview of all service-related processes and components involved from starting the client to ending the service session. Three different cases have been specified with regard to the order of processes and the relation of the components involved:

- The user listens to SoftMix Radio
- The user uploads content to the network for others to use
- The user turns on "Broadcast mode", i.e. publishes a playlist that may involve content uploaded by the broadcasting peer as well as content by other users that is available in the network.

These three cases partly involve different components and processes. Therefore, they will be described individually in order to avoid confusion.

3.2.4.1 Case: User listens to SoftMix Radio

1. As part of the registration procedure the user will select a pre-defined profile
2. At the start the Client connects with the Profile Engine.
3. The Profile Engine connects with the Recommendation Engine.
4. The Recommendation Engine checks the relevant Programme Frame and filters the Content Search accordingly.
5. The Recommendation Engine checks the network for available media (Search Engine).
6. The Recommendation Engine selects matching files and generates a playlist by filling them into the Programme Frame.
7. To play content the Client connects with the Conditional Access Management Subarchitecture. The client will use the specified sequence of SIP MESSAGE's using the CPS's text-based protocol to negotiate a license for the content. The parameters in the initial request will be:
   - URI of user requesting the content license (obviously)
   - Content Identifier
   - Content Provider Identifier (to prevent a content provider from rebundling anothers content when unauthorised to do so)
   - Optional Fair-Usage parameter (educational, personal, commercial etc.)
8. The Client checks the network for the file version/instance that best suits its device characteristics.
9. The client is returned the URI for the correct content overlay associated with the requested content and the content is streamed to Client

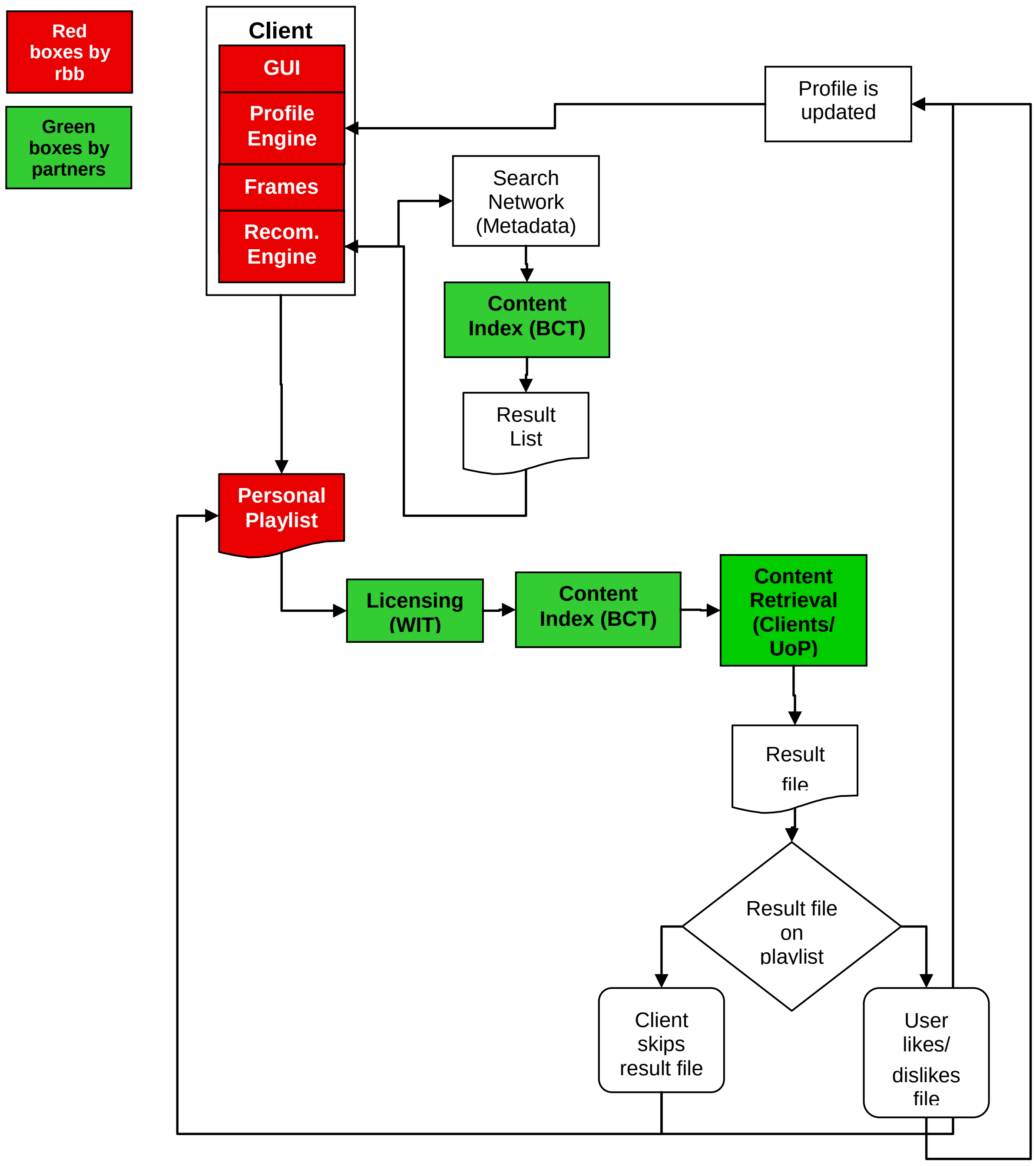


Figure 12: Overview Diagram / Flow Chart

3.2.4.2 Case: User publishes content

For content publishing there are generally two options

1. Content can be published by a "simple" peer
2. Content can be published by a professional or semi-professional content provider by means of a "super peer".

The main difference lies in the way metadata are connected with content source files.

- Any "simple" peer would start by offering a file to other peers and would have to add metadata in a dedicated user interface which interfaces to a central database. Content Providers like rbb will provide already available metadata and source files by interfacing with their CMS and may copy this information to their super peer to provide these sources to the network.

Apart from this difference, the procedure is identical:

1. The publishing peer starts the publishing procedure
2. Metadata are being added to the content item(s) to be published or rather the URIs of the source files are being added to the metadata sets.
3. The Transcoding Service converts the content file into instances that are connected with the same metadata.
4. All instances are being encrypted according to a number of presets.
5. The metadata set is being added with the URIs of the new instances accordingly so that the search result would point to all available instances at a time. Furthermore, technical information on the instances is being added, such as

- bit rate, encoding mechanism, etc.;
- URI for the overlay which can distribute the content,
- Encoding information,
- Content Provider Id (provided earlier by the Content Protection Subsystem),
- Content Licensing Business rules,
- Content Charging Information

6. Finally, the completed metadata file is being stored in the Content Index

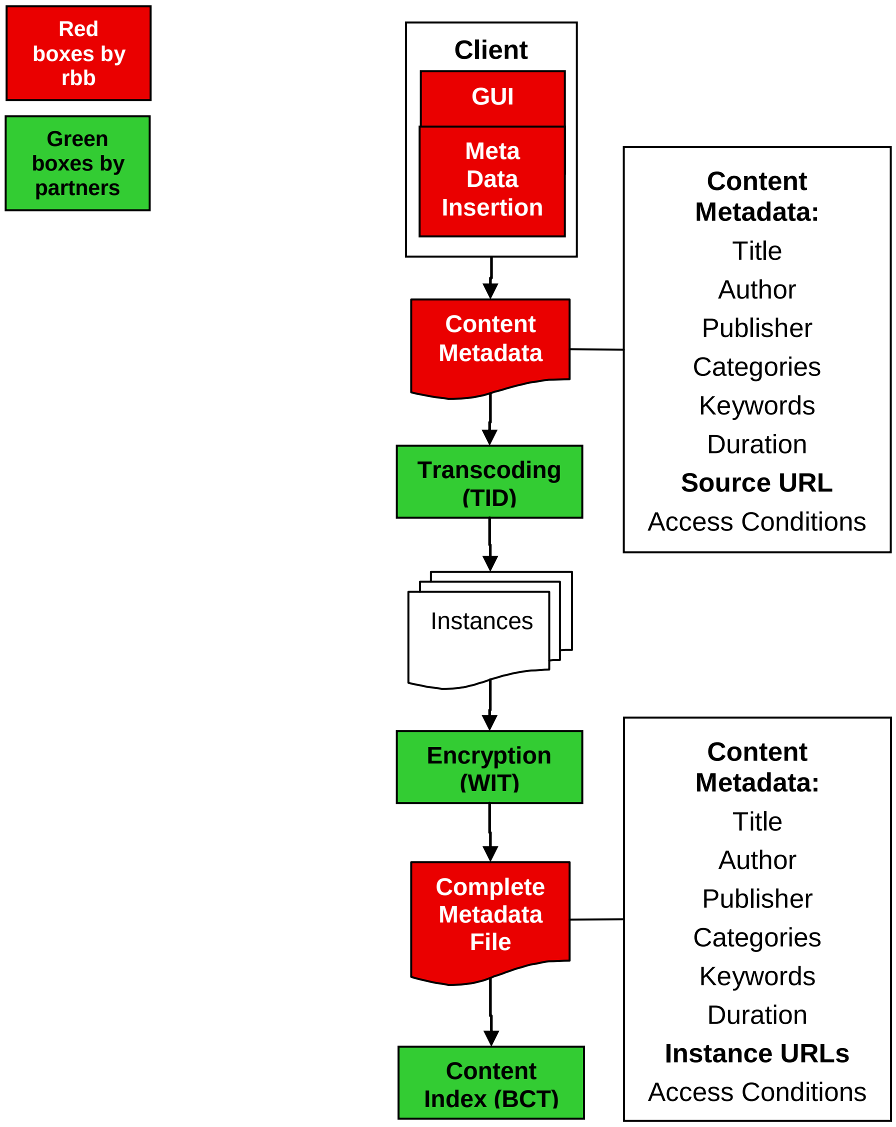


Figure 13: Overview Content Publishing Procedure

3.2.4.3 Case: User turns on "Broadcast Mode"

This third case is a special case of content publishing.

In a way, SoftMix's Broadcast Mode is a Social App which enables users to share their preferred files with others. These preferred files may be self-created, user-generated content as well as a mere playlist of content already available on the network or a mix of both. For example, the user on broadcast mode could put together a playlist with his/her favourite word or music files and add his/her own moderating sequences to guide the listener through their programme.

Thus, Broadcast Mode requires both publishing facilities as well as features to generate playlists. This is an optional feature that would also require clarification of various legal issues but would certainly enrich the use of SoftMix for the Generation Web2.0.

4 Summary

This document described all content-related services and processes, explaining in detail what they are used and needed for. The combination of the components explains why and how P2P and SMS technologies work together hand in hand to enable and support the ambitious objectives of the project's main demonstration service. While content distribution mainly focuses on P2P technologies, Transcoding and Conditional Access Management are building upon IMS features in order to guarantee privacy and Quality of Service.

SoftMix, the VITAL++ demonstration service integrates all of these features in order to provide the best possible service based on the features of the project's architecture.

- End of document -