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	**Peer-to-Peer client evaluation and market overview**

Instrument:	STREP
Theme:	ICT-2-1.6
Report Due:	M4
Report Delivered:	M5
Lead Contractor for this deliverable:	CTRC
Contributors to this deliverable:	E. Pallis (CTRC), E. Markakis (CTRC), A. Sideris (CTRC), Odysseas Koufopavlou (UoP), Charalabos Skianis (UoP), Nikolaos Efthymiopoulos (UoP), Nico de Abreu (RBB), Bjoern Stockleben (RBB), Jens Fiedler (FOKUS), Christian Riede (FOKUS), George Karidis (BCT), George Kapelios (VoG), Shane Dempsey (WIT), Jose Luis Pena (TID), Juana Sanchez (TID).
Estimated person Months:	7
Start date of project:	1st June 2008
Project duration	30 months
Revision:	Version 1.0
Dissemination Level:	PU - Public

This page intentionally blank

1 Table of Contents

2 List of Figures

3 Document History

The aim of this deliverable is to provide relevant information regarding the Peer-to-Peer client evaluation and market overview

Revision Month	Filename version	Summary of Changes
M1	V0.9	Report
M1	V1.0	Template corrections
M2	V1.1.(x)	State-of-the-art and business cases
M3	V2.1.(x)	Criteria and requirements
M4	V2.2.(x)	P2P Clients evaluation
M4	V2.3.(x)	P2P and IMS clients evaluation
M4	V2.4.(x)	P2P and IMS clients evaluation
M4	V2.5.(x)	P2P and IMS clients evaluation
M4	V3.1.(x)	Corrections and amendments
M5	V3.2.(x)	Refinements and discussion on the entire D2.1
M5	Final	Finalisation of D2.1

4 Executive summary

This “VITAL++” deliverable (D2.1) – “Peer-to-Peer client evaluation and market overview”, part of Workpackage 2 (WP2), Task 2.1 – elaborates on the multitude of existing Peer-to-Peer Software Clients, with special focus on their feature set. It initially studies and analyses a number of usage scenarios, for deriving the criteria set, according to which existing P2P Client evaluation is carried out. Special focus is set on the ability of P2P clients to be enhanced/reused within IMS networks. The deliverable concludes by identifying the key features required to support content distribution in an IMS environment, and which will be used prior to the selection of the P2P clients that will be used in WP3.

5 Introduction

This document is focused on the market overview of existing P2P Clients, by evaluating and analysing their feature set, with special focus on their ability to be enhanced/reused within IMS networks. Based on the results of the evaluation process, a set of features for P2P clients is identified suitable for supporting content distribution and meeting the operators' requirements as they stem from a number of use cases. Eventually, some of the evaluated P2P clients will form the basis for feature adaptation of the client selected and used in Work Package3.

More specifically, the deliverable initially elaborates on the current state-of-the-art in P2P systems and IMS technology (Section 6), by studying and analysing architectures, topologies and delivery mechanisms according to three distinct use-cases: a) content (file) distribution , b) peer-to-peer assisted video on demand, and c) peer-to-peer live streaming. Following these use-cases, Section 7 elaborates on three representative scenarios that depict/reflect the vision of VITAL++ from a service provider point-of-view, including remote access to content, personalised content access (SoftRadio), as well as live streaming of content provided by different service providers and remote access to content from rural areas. Based on these scenarios, the deliverable studies (in Section 8) the criteria according to which P2P Client evaluation will be carried out, by elaborating on the reason for evaluation, the basis upon which the P2P clients will be evaluated, and by identifying the criteria and requirements for IMS operation. Following these criteria/requirements, Section 9 presents the evaluation of existing P2P clients, categorised according to their use-case, i.e. for the content distribution, VoD and live-streaming scenario.

Similarly, Section 10 presents two IMS clients contributed by two partners, and argues about the extensions that need to be designed in the context of this project. The first client is the IMS one built as part of a predecessor EU project, called VITAL, while the second client is the Monster IMS one, part of the OPEN IMS framework.

Finally Section 11 elaborates on the overall evaluation and design goals for P2P and IMS clients to be adopted by VITAL++, while Section 12 concludes this deliverable by summarising the findings of this market overview.

6 State of the art

In this chapter, we will reflect the ongoing work on the main technologies, which form the base of the VITAL++ project. These are P2P services delivery methods (including the content distribution, the live streaming and video on demand), the IP multimedia subsystem (IMS), and actual related work for combining those technologies.

6.1 State-of-the-art in P2P

6.1.1 Content Distribution

Content distribution is the most well understood P2P service from the scientific community. In content distribution a server contains large amount of data and they must be transferred to a vast number of peers. Files are cut into blocks and each block is delivered in a small subset of peers. Peers exchange blocks until they retrieve the whole content. There are four major issues that we have to address in order to develop an efficient content distribution system.

The major factor that affects the performance is content bottleneck. This is the case in which peers haven't different blocks to exchange and their upload bandwidth remains idle. To efficiently download the file, it is important to design the file-sharing protocol such that each peer is matched with others who have the pieces of the file that it needs and further, to ensure that the downloading bandwidth of each peer is fully utilized.Another promising approach towards the solution of this problem is network coding[1]. In Network coding each peer creates a linear combination from many blocks and transmits this combination instead of single blocks. When a peer receives enough blocks to decode them performs this action and regains the original blocks.

The second factor is the rate in the change in the number of peers in the system. Therefore, it is useful to study how the number of peers evolves as a function of the request arrival rate, the peer departure rate, the uploading/downloading bandwidth of each peer, etc.

The third factor is the development of incentives to prevent free-riding: Free-riding is a major cause for concern in P2P networks. Free-riders are peers who try to download from others while not contributing to the network, i.e., by not uploading to others. Thus, most P2P networks try to build in some incentives to deter peers from free-riding. Once the incentive mechanism is introduced into the network, each peer may try to maximize its own net benefit within the constraints of the incentive mechanism. Thus, it is important to study the effect of such behavior on the network performance.

[1] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. Infocom, 2005.

At last but not least scalability is a major concern and the motivation of the utilization of P2P systems: To realize the advantages of P2P file sharing, it is important for the network performance to not deteriorate, and preferably to actually improve, as the size of the network increases.

In order to **minimize the load that P2P content distribution introduces to the network provider** there are three useful techniques:

- the first is a locality aware overlay[2], where peers exchange blocks and metadata with the closest in the underlying physical network, and
- the second is caching of data in locations that they requested with high frequency[3].
- the third is the avoidance of network paths that have high cost according to the ISP contracts

Earlier approaches[4] have shown that cooperation between IMS and a P2P content distribution system (BitTorrent) is possible, but must be secured with appropriate DRM techniques. Nevertheless, a smooth integration into the IMS architecture is still far from being achieved.

6.1.2 Peer-assisted video on demand (VoD)

In Peer-assisted video on demand (VoD), the peers that are viewing the publisher's videos also assist in redistributing the videos. Since peer-assisted VoD can move a significant fraction of the uploading from the server to unused resources (e.g. bandwidth) of the peers, it can potentially dramatically reduce the publisher's bandwidth costs. As the aggregate bandwidth in these systems varies[5] there are three modes in which the systems works:

- the surplus mode where the provided upload bandwidth from peers exceeds the demand,
- the balance mode where both are equal, and
- the deficit mode where the provided upload bandwidth is less than the demand.

There are many proposed techniques for bandwidth allocation in VoD[6]. Due to the different parts of the video that requested and the various locations that they are located a wise policy for resource allocation must be applied in order

[2] http://www.wcl.ece.upatras.gr/index.php?sid=&lang=el&id=51

[3] Jussi Kangasharju, Keith W. Ross, David A. Turner, Optimizing File Availability in Peer-to-Peer Content Distribution, Infocom 2007.

[4] J. Fiedler, T. Magedanz, A. Menendez: "IMS secured content delivery over peer-to-peer networks", Proceedings of SIGMAP 2007, Spain, July 28-31, 2007, INSTICC Press, Portugal, p. 5-12, ISBN 978-989-8111-13-5.

[5] Cheng Huang, Jin Li, Keith W. Ross, Can video on demand be profitable? , SIGCOMM 2007.

[6] Siddhartha Annapuredd, Saikat Guha, Christos Gkantsidis Is HighQuality VoD Feasible using P2P Swarming?, WWW 2007.

to assign and schedule uploads to peers. Pre-fetching is applied and was found to be a promising approach towards this goal. The last critical issue is overlay topology management where algorithms for peer matching and optimization applied. It has been shown that overlay architecture greatly affects the performance of peer assisted VoD.

On the other hand to the best our knowledge there are many unsolved issues towards the creation of a Peer-assisted video on demand streaming system, thus, there is a need to develop such a system starting first from the development of a live streaming system and its expansion.

6.1.3 P2P live streaming

P2P streaming is a real time application with strict delivery time constraints and very demanding in terms of the aggregate bandwidth required for the delivery of the stream to the participating peers. In general, a server generates a video stream at a given service rate which is then divided into blocks followed by their delivery to a small subset among the participating peers. As a final step, all peers exchange these blocks in order to reproduce the whole video stream.

Peers involved in these systems, may have heterogeneous upload bandwidth capabilities while the average upload bandwidth capability of the participating peers constrains the maximum service rate of the video stream that can be delivered successfully to all peers[7]. An efficient P2P streaming system must be able to deliver a video stream with service rate as close as possible to the average upload capability of the participating peers with the smallest possible delay, called *setup time*. With the term setup time we define the time interval between the generation of a block from the origin server and its distribution to every peer in the system.

Furthermore, a P2P live streaming system has to adapt to the dynamic underlying network conditions and cope with dynamic node arrivals and departures. This results in varying number of peers and uploading capacities which impact the stability of the system with respect to the uninterrupted delivery of the streaming service. Finally, fairness among nodes guarantees equal bandwidth distribution to the participating nodes and so they acquire equal number of blocks of the video stream in the predefined setup time.

Several approaches that have been recently proposed for creating P2P streaming systems may fall into two categories.

The first is based on a formation of forests of trees whereby each node is a leaf in every tree but one. Blocks are assigned equiprobably into a number of

[7] Kumar R., Liu Y., Ross K. W., Stochastic Fluid Theory for P2P Streaming Systems, In 26th IEEE International Conference on Computer Communications (INFOCOM), pp. 919-927. IEEE Press, Anchorage (2007).

stripes equal to the number of the formed trees. Each tree distributes (pushes) one stripe by propagating each one of its blocks from parent to its children. In this category blocks are pushed according to the overlay topology. SplitStream is a distributed implementation of this approach that is based on a locality aware DHT called Pastry[8]. SplitStream and systems alike have the advantage of being topologically aware (trees are formed according to the network distance between nodes) leading to small setup time as the propagation of a block from the root of the tree to the leaf nodes is done through nodes which are physically close in the underlying network. However these systems suffer from two main drawbacks: a) they don't take into account the heterogeneous upload capacities of the peers[9], and b) they can't cope up with the dynamic behavior of the participating peers as well as the underlying network as observed in commercial P2P streaming systems[10,11,12]. When a peer leaves the overlay, the path between it and its descendants is broken resulting in idle descendants during the reconstruction phase of the tree.

In the second category[13,14], each node maintains connections with a relatively small number of nodes which are considered as its neighbors in the overlay. The overlay is constructed randomly or according to the upload capacities of the nodes that participate in it. Blocks that are generated by a server have playback deadlines. Each peer exchanges and maintains a number of lists (buffers), one per neighbor. Each one of these buffers contains those blocks of its neighbor that their playback deadline has not expired yet. To this end, a peer is capable at any time of making a decision about which block should be transmitted to which neighbor. This decision process is implemented by a scheduler running in every node. The characteristic of these systems is that the block transmissions are agnostic to the overlay topology.

Due to their architecture the main advantage of them is their flexibility which allows them to take advantage of the heterogeneity of the participating peers

[8] Rowstron A., Druschel P., Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems,In 18th IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, (2001).

[9] Magharei N., Rejaie R., Guo Y., Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches, In 26th IEEE International Conference on Computer Communications (INFOCOM), pp. 1424-1432. IEEE Press, Anchorage (2007).

[10] X. Hei, C. Liang, J. Liang, Y. Liu and K.W. Ross, A Measurement Study of a Large-Scale P2P IPTV System, IEEE Transactions on Multimedia Volume: 9, Issue: 8 pp. 1672-1687 (2006).

[11] X. Hei, C. Liang, J. Liang, Y. Liu and K.W. Ross, A Measurement Study of a Large-Scale P2P IPTV System, November 2006, to appear in IEEE Transactions on Multimedia.

[12] PPLive http://www.pplive.com.

[13] Massoulie L., Twigg A., Gkantsidis C., Rodriguez P., Randomized decentralized broadcasting algorithms, In 26th IEEE International Conference on Computer Communications (INFOCOM), pp. 1073—1081. IEEE Press, Anchorage (2007).

[14] Magharei N., Rejaie R., Guo Y., Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches, In 26th IEEE International Conference on Computer Communications (INFOCOM), pp. 1424-1432. IEEE Press, Anchorage (2007).

and deal with the dynamic behavior of the system leading to higher levels of bandwidth utilization. However, these systems can't exploit the network proximity among the peers that exchange blocks. This means that the time required for a block to be transferred from one node to another and hence the required time for all nodes to acquire the block (setup-time) could reduced if the overlay exploits the locality between peers. Another drawback of overlays agnostic to locality is that buffer exchanges between neighbors performed with high network latency. This effect leads duplicate block transmissions and so to wasted upload bandwidth.

6.2 The IP multimedia subsystem (IMS)

The IMS is an architectural framework for delivering IP-multimedia to mobile users. It was originally designed by the wireless standards body 3rd Generation Partnership Project (3GPP), and was intended to lead a way for mobile networks beyond GSM. Its original formulation (3GPP R5) represented an approach to delivering "Internet services" over GPRS. This vision was later updated by 3GPP, TISPAN by requiring support of networks other than GPRS, such as Wireless LAN and fixed line.

The IMS follows a three tiered architecture: Application Layer, Call Control Layer and Transport Layer. The Application Layer provides an independent service layer for the execution of value-added services and content. The Call Control Layer bases on advanced signalling protocols and is arranged in softswitches or session servers. The Transport Layer consists of dedicated nodes, so called Media Gateways. They work as routers in the classical IP fashion and process content data controlled by the Call Control Layer. The IMS layered architecture is depicted in the following illustration (Figure 1).

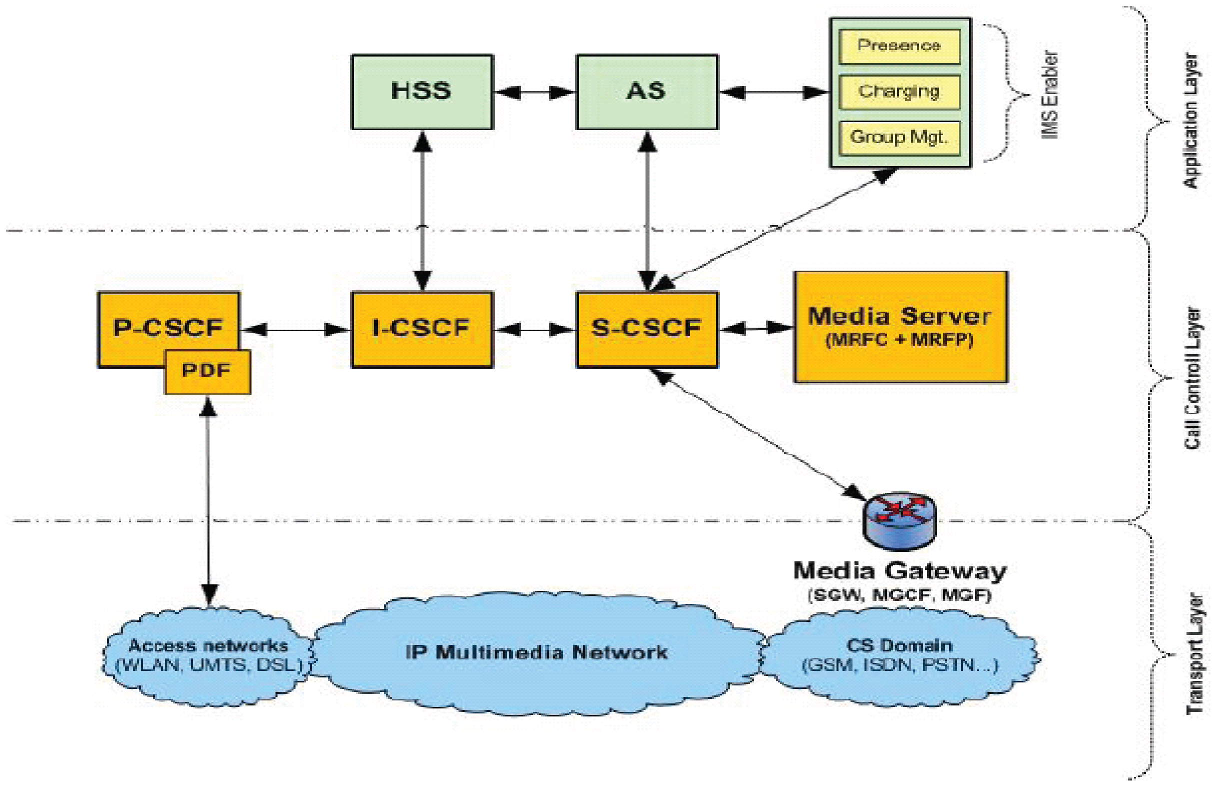


Figure 1 IMS Layered Architecture

The IMS core network system consists of different functions, interacting over standardized interfaces (reference points), which form one IMS administrative network. A function is not necessarily identical to a node (hardware box): an implementer is free to combine 2 or more functions in one single node, or to spread a single function over multiple nodes. Each function can also be present multiple times in a single network, for load balancing, availability purposes or organizational issues. Reference points are realized by standardized protocols, like SIP or DIAMETER. Figure 2 illustrates the most relevant IMS functions of the core network and the related reference points between them.

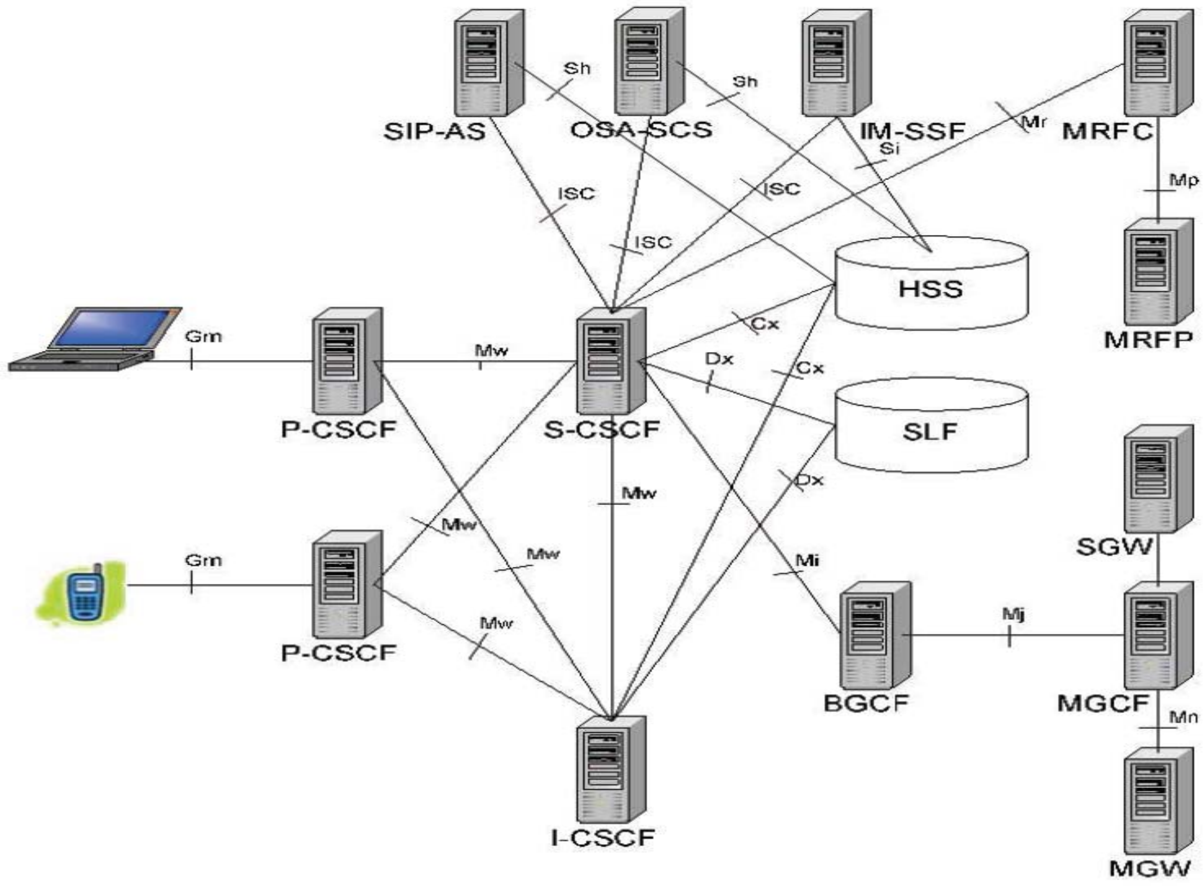


Figure 2 IMS Functions and Reference Points

The Home Subscriber Server (HSS) is the master database for a given user. It is the entity containing the subscription-related information to support the network entities actually handling calls/sessions. The HSS is responsible for holding user related information as:

- User Identification: Numbering and addressing information
- User Security information: Network access control information for authentication and authorization
- User Location information at inter-system level: the HSS supports the user registration, and stores inter-system location information, etc.
- User profile information: E.g. subscribed services, etc.

The Application Server (AS) is an IMS entity that hosts and executes IP multimedia services. The AS is the “expansion slot” for an IMS network. Here, 3rd party products and services are located. The AS can operate in three different modes:

- SIP proxy mode
- SIP user agent (UA)
- SIP back-to-back-user-agent (B2BUA)

IMS Enabler IMS enabler are special ASs with generic functions which perform functionalities like Presence, Group Management or Charging.

The Policy Decision Function (PDF) is responsible for making policy decisions based on session and media-related information obtained from the P-CSCF. Policy decisions refers in this case to QoS control.

The Proxy-Call Session Control Function (P-CSCF) is the rst contact point for users within the IMS. All SIP signaling trac from or to the UE goes via the P-CSCF. The P-CSCF validates the request, forwards it to selected destinations and processes and forwards the response.

The Interrogating-CSCF (I-CSCF) is a contact point within an operator's network for all connections destined to a subscriber of that network operator. The I-CSCF contacts the HSS to obtain the name of the S-CSCF that is serving a user and forwarding a SIP request or response to the S-CSCF. The I-CSCF provides a hiding functionality. The I-CSCF may contain functionality called the Topology Hiding Inter-network Gateway (THIG). THIG could be used to hide the conguration, capacity and topology of the network from outside an operator's network

The Serving-CSCF (S-CSCF) is the heart of the IMS. It is located in the home network and performs session control and registration services for UEs. While UE is engaged in a session the S-CSCF maintains a session state and interacts with service platforms and charging functions as needed by the network operator for support of the services. There may be multiple S-CSCFs, and S-CSCFs may have dierent functionalities within an operator's network.

MS The Media Server (MS) is an IMS entity that consists of functional components: Multimedia Resource Function Controller (MRFC) and The Multimedia Resource Function Processor (MRFP).

The MRFC is needed to support bearer related services, such as conferencing, announcements to a user or bearer transcoding.

The MRFP provides user-plane resources that are requested and instructed by the MRFC. The MRFP performs the following functions:

- Mixing of incoming media streams (e.g., for multiple parties)
- Media stream source (for multimedia announcements)
- Media stream processing (e.g., audio transcoding, media analysis)

The Media Gateway consists of three essential IMS components:

- Media Gateway Control Function (MGCF)
- Signaling Gateway (SGW)
- Multimedia Gateway Function (MGF).

So the Media Gateway enables communication between IMS and circuit switched (CS) users.

6.2.1 3GPP IMS Releases

IMS was first standardized in 3GPP Release 5. It then experienced multiple extensions, expansions, updates and spinoffs. Figure 3 gives an overview of the evolution of IMS throughout all the standardization bodies.

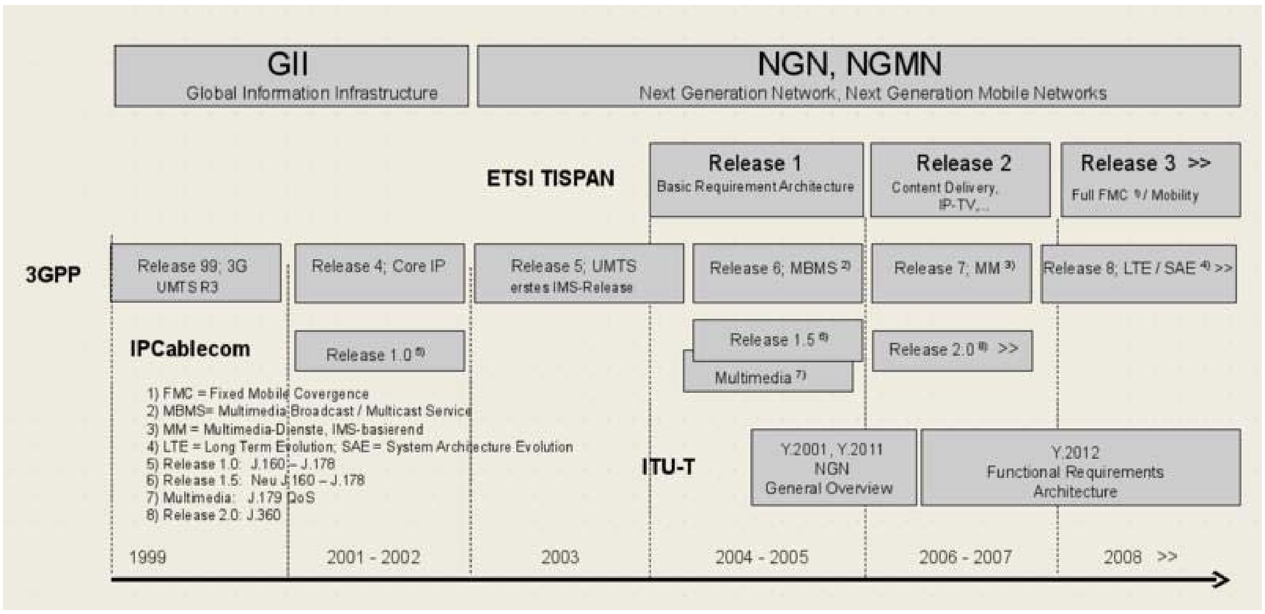

GII
Global Information Infrastructure
NGN, NGMN
Next Generation Network, Next Generation Mobile Networks
ETSI TISPAN
Release 1
Basic Requirement Architecture
Release 2
Content Delivery, IP-TV,...
Release 3 >>
Full FMC 1) / Mobility
3GPP
Release 99; 3G UMTS R3
Release 4; Core IP
Release 5; UMTS erstes IMS-Release
Release 6; MBMS 2)
Release 7; MM 3)
Release 8; LTE / SAE 4) >>
IPCablecom
Release 1.0 5)
Release 1.5 6)
Multimedia 7)
Release 2.0 8) >>
1) FMC = Fixed Mobile Covergence
2) MBMS= Multimedia Broadcast / Multicast Service
3) MM = Multimedia-Dienste, IMS-basierend
4) LTE = Long Term Evolution; SAE = System Architecture Evolution
5) Release 1.0: J.160 – J.178
6) Release 1.5: Neu J.160 – J.178
7) Multimedia: J.179 QoS
8) Release 2.0: J.360
ITU-T
Y.2001, Y.2011 NGN General Overview
Y.2012 Functional Requirements Architecture
1999
2001 - 2002
2003
2004 - 2005
2006 - 2007
2008 >>

Figure 3 IMS Timeline

The following table gives an overview of the different 3GPP IMS releases and their related features.

Release 5	• VoIP, IM, Presence support on top of GPRS • IMS Architecture: IMS Architecture, network entities, reference points (interfaces) between the network entities. • User Identities: Public/Private User Identity, usage of the SIP-URI and TEL-URI, ISIM, the use of the USIM instead of the ISIM. • IMS Session Control: o IMS Registration o IMS Session Routing o Session- Modification and Teardown o SIP Signaling Compression • IMS Service Control:

	o invocation/control of IMS Application Servers based on Filter Criteria in the CSCF o IM-SSF and there-use of CAMEL Services o Interconnect with the OSA-GW and the use of OSA services • QoS Mechanisms: o QoS Preconditions o QoS/Media Authorization based on the PDF • Security Mechanisms: o IMS User Authentication o Message Integrity Protection, o IMS Network Domain Security
Release 6	• QoS, PoC support • IMS SIM cards • IPv6 deployment • IMS Interworking: o With the CS-Domain (more details for CS and PSTN) o With SIP Clients in the Internet (IPv4/v6 Interworking) o WLAN access to the IMS (not completed) • IMS Session Control: o multiple registrations o routing of group identities • Security Mechanisms: o confidentiality protection of SIP messages o use of public key infrastructure o Ut-interface security o early IMS security • IMS Services: o Presence o Instant Messaging o Conferencing o Group management
Release 7	• Identification of Communication Services in IMS • Supporting Globally Routable User Agent URIs in IMS • IMS Support of Conferencing and Messaging Group

	Management • Location Services enhancements (LCS3) • Advanced Global Navigation Satellite System (A-GNSS) concept (LCS3-AGNSS) • Enhancements for fixed broadband access to IMS • Access Class Barring and Overload Protection • Protocol-related new Features o DIAMETER on the GGSN Gi interface o DIAMETER on the PDG Wi inteface • Support of SMS over generic 3GPP IP access • Dynamic and Interactive Multimedia Scenes (DIMS) • Personal Network Management (PNM) • WLAN-UMTS Interworking Phase 2
Release 8	• SAE for LTE access • InterWorking Function (IWF) between MAP based and Diameter based interfaces • Flexible Alerting • Support of Packet Cable access • corporate network access • Interworking between User-to-User Signalling (UUS) and SIP • Earthquake and Tsunami Warning System • Customized Alerting Tone (CAT) Service • Value-Added Services for Short Message Service • 3G Long Term Evolution - Evolved Packet System (RAN)
Release 9	• Services Alignment and Migration • Registration in Densely-populated area (RED) • End-User Identity • Public Warning System • Support of Personal Area Networks (PAN) • User Data Convergence • Protection against Unsolicited Communication for IMS (PUCI) • Machine-type Communications

Table 1 3GPP IMS Releases - Feature List

6.2.2 ETSI TISPAN

TISPAN is the ETSI core competence centre for fixed networks and for migration from switched circuit networks to packet-based networks with an architecture that can serve in both to create the Next Generation Network.

Building upon the work already done by 3GPP in creating the SIP-based IMS (IP Multimedia Subsystem), TISPAN and 3GPP are now working together to define a harmonized IMS-centric core for both wireless and wireline networks.

This harmonized ALL-IP network has the potential to provide a completely new telecom business model for both fixed and mobile network operators. Access independent IMS will be a key enabler for fixed/mobile convergence, reducing network installation and maintenance costs, and allowing new services to be rapidly developed and deployed to satisfy new market demands.

TISPAN considers effective cooperation with external bodies as essential to the coordination of the global message and further globalization of the TISPAN NGN product.

6.2.2.1 Release 1

NGN Release 1 was launched by TISPAN in December 2005, providing the robust and open standards that industry can use as a reliable basis for the development and implementation of the first generation of NGN systems. The addressed features are:

- Overall NGN Stage 1&2
- Network Attachment Subsystem (NASS)
- Resource and Admission Control Subsystem (RACS)
- PSTN/ISDN Emulation Subsystem (PES)
- PSTN/ISDN Simulation Subsystem (PSS)
- IMS-simulated PSTN/ISDN Supplementary Services (PSS)
  - Videotelephony over NGN
  - Emergency services
- IMS-specific Supplementary Services (ISS)
  - Presence Service (Presence)
  - IMS Messaging
- Interworking NGN – CS networks – IP networks – IMS
- NGN management

6.2.2.2 Release 2

TISPAN has published Release 2, with a focus on enhanced mobility, new services and content delivery with improved security and network management. The addressed features in Release 2 are:

- IMS-specific Supplementary Services (ISS)
    - SMS over NGN IMS
    - Direct Communication (DC) Service
- IP Television (IPTV)
- Fixed Mobile Convergence (FMC)
- Corporate Network
- Customer Network Gateway (CNG)
- Overload and Congestion Control (OCC)
- NGN Subscription Management (SM)

6.2.2.3 Release 3

Currently, ETSI TISPAN is working on release 3 of the specifications and standards. As this work is quite new, only enhancements and improvements of already existing features from releases 1 and 2 are on the workplan.

6.3 State-of-the-art in IMS and P2P cooperation

Ongoing work for P2P and IMS cooperation and integration mainly focuses on the signalling level. This means that mainly external or client based DHTs are used for backup purposes for a centralistic system in case of its failure.

The SIP, the core signalling protocol of IMS, is of interest for P2P overlays. At the IETF, a working group has been established, which defines methods and procedure to use SIP for constructing P2P overlays. This SIP extension is called P2PSIP[15].

One approach for P2P and centralistic interworking is the CoSIP, developed at the University of Tübingen[16]. In their solution, they introduce a CoSIP proxy server, which replaces the traditional SIP proxy in a fashion, that it can chose either to use a centralistic architecture or in case of failure, a P2P based DHT. Here, the DHT is not necessarily build by the SIP clients, as one of the design goals was to keep these unmodified. This solution does not explicitly focus on IMS, but gives a good example, on how an originally centralistic service can benefit from a P2P system.

[15] D. Bryan, P. Matthews, E. Shim, D. Willis, and S. Dawkins, “Concepts and Terminology for Peer to Peer SIP”, July 2008, draft-ietf-P2Psip-concepts-02 (work in progress).

[16] A. Fessi, H. Niedermayer, H. Kinkelin, G. Carle: ”A Cooperative SIP Infrastructure for Highly Reliable Telecommunication Services“, IPTComm 2007, PRINCIPLES, SYSTEMS AND APPLICATIONS OF IP.

A similar, but more advanced approach was done by Morocco[17]. This takes IMS and its core components into account, but also uses a P2PSIP network for message routing. Clients, which support only the P2PSIP, can communicate over a proxy-peer with IMS clients. Thus, this solution describes rather the interworking of non-hybrid clients, belonging to exactly one of the interworking schemes (IMS or P2PSIP).

An example, where the IMS location service has been used in a P2P application is a mobile chess application, developed at the University of Aachen[18]. Also the use of IMS service enablers has been proposed for Mobile-TV[19].

It has already been understood, that cooperation must not only go in one direction. As pure P2P networks lack authentication, this is one option to enhance P2P with properties of centralistic systems. The work of Cao, Bryan and Lowekamp introduces a group of trusted authentication servers, which provide a login service and key management[20].

One approach, which also focuses on media delivery, was a case study done by Fraunhofer FOKUS, where they extended an existing IMS network (namely the Open IMS Playground), with a P2P application server and an external DRM server[21]. The P2P-AS had the purpose to coordinate BitTorrent swarms (content) with DRM licenses. A Java IMS client has been modified to use external tools to process encrypted content. The major drawback of this solution was the fact that many components (e.g. the DRM-server) were not fully integrated in the IMS architecture and used protocols, not standardized by the IETF and thus no standardized reference points.

Finally, cooperation of P2P and IMS has reached the standardization bodies. At the ETSI, a Work Item has been established for this field of technology. The work item is titled “Peer-to-peer for content delivery for IPTV services: analysis of mechanisms and NGN impacts”. This Work Item is going to be developed in WG1/2 (Requirements and Architecture) but liaisons with other groups are not discarded.

[17] Enrico Marocco, "Interworking between P2PSIP Overlays and IMS Networks - Scenarios and Technical Solutions", ICIN 2007.

[18] Guido Gehlen, Fahad Aijaz, Yi Zhu, Bernhard Walke, “Mobile P2P Web ervices using SIP”, Journal Mobile Information Systems, IOS Press, ISSN 1574-017x, Issue Volume 3, Number 3-4 / 2007, Pages 165-185.

[19] Raimund Schatz, Siegfried Wagner, Norbert Jordan, "Mobile Social TV: Extending DVB-H Services with P2P-Interaction," Digital Telecommunications, International Conference on, vol. 0, no. 0, pp. 14, Second International Conference on Digital Telecommunications (ICDT'07), 2007.

[20] Feng Cao, David A. Bryan, and Bruce B. Lowekamp, “Providing Secure Services in Peer-to-Peer Communications Networks with Central Security Servers” Proceedings of the 2006 International Conference on Internet and Web Applications and Services (ICIW'06), February 2006.

[21] J. Fiedler, T. Magedanz, A. Menendez: “IMS secured content delivery over peer-to-peer networks”, Proceedings of SIGMAP 2007, Spain, July 28-31, 2007, INSTICC Press, Portugal, p. 5-12, ISBN 978-989-8111-13-5.

Concluding from the previous, it is obvious that IMS and P2P interworking is a hot topic not only for academic research, but also for standardization and thus for industry.

7 Usage scenarios

7.1 Scenario 1: RBB remote

Rbb remote is a service concept that makes rbb content available to legitimate subscribers/viewers who happen to be outside the broadcast region for a time (travelling, studying, etc.). This extra content provision, however, must not cause any extra costs to the broadcaster (in this case rbb), according to the current legal regulations.

7.1.1 Content Access outside a geo-blocked area

Due to licensing policies AV Content on the internet is often geo-blocked and thus 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.

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.

7.1.2 Out-Of-Area content distribution

Content distribution via Internet is affordable for larger content providers without relying on P2P services. Public broadcasters such as Germany's ZDF offer up to 50.000 concurrent TV quality direct video streams and are ready to scale as demand rises. Yet the service area of the public broadcasters is legally limited to their home country. However, viewers from abroad might also be interested in the content, e.g. because they are speaking the same language. However, if demand from abroad rises there is no according re-financing of distribution costs, as those viewers do not pay TV license fees to the respective broadcaster. As a solution, broadcasters could offer direct streams only in their own country and rely on P2P outside. This way nobody would be excluded, still funds would be used according to the national contracts.

Technically, users would have to register via the IMS in order to get access to the content in the appropriate way. Users might even explicitly offer their resources to support other network users. Thus a decentralised distribution system could develop which could even “sell” its resources to other users.

7.1.2.1 Multimedia conferencing

Modern working patterns and collaboration models dictate the necessity for people located in different, whether close or distant, geographic sites to work together in a manner quite similar to being in the same room. Multimedia conferencing provides the generic solution to this need. However, restrictions might be encountered, mainly attributable to the bandwidth available to each of the participants in relation to the location and service provider characteristics of the rest. In addition, conferencing places wider requirements to the available services than mere real-time AV; whiteboarding and instant file exchange are also needed.

7.2 Scenario 2: SoftRadio – Personalised Radio Experience

7.2.1 Mixing programmes from different sources

The VITAL++ platform may offer an opportunity for independent service providers to bundle content from different sources in order to provide exciting programmes for TV and radio services. Likewise users could mix their own programme from a variety of content.

7.2.2 End user Service – The *Softradio* concept

Softradio is a concept that enables the users to mix their own radio programme from a multitude of content sources (audio and video) by one or multiple content providers. Still the most important characteristics of a broadcast programme including a clear branding are preserved.

In rbb's case, the user would be able to choose from different mixing sources, thereby utilising the three VITAL++ features: streaming, on demand content and file sharing:

7.2.2.1 Streaming

- The classical linear radio programme
- Loop streams of the nightly music programmes

7.2.2.2 On Demand

- Audio content from the community platform “meinfritz.de”
- “Fritz unsigned” music content of newcomer bands from the Berlin-Brandenburg area
- Podcast content (audio and video) such as news, comedy, etc.
- Video content from the rbb storage.

7.2.2.3 File Sharing

- Audio content (music, etc.) from own PC and from buddies’

The linear programme often follows a well-structured schedule in form of a “radio clock” (see Figure 4), which shows fixed slots for news, sports, service, music, etc. (see Figure 5).

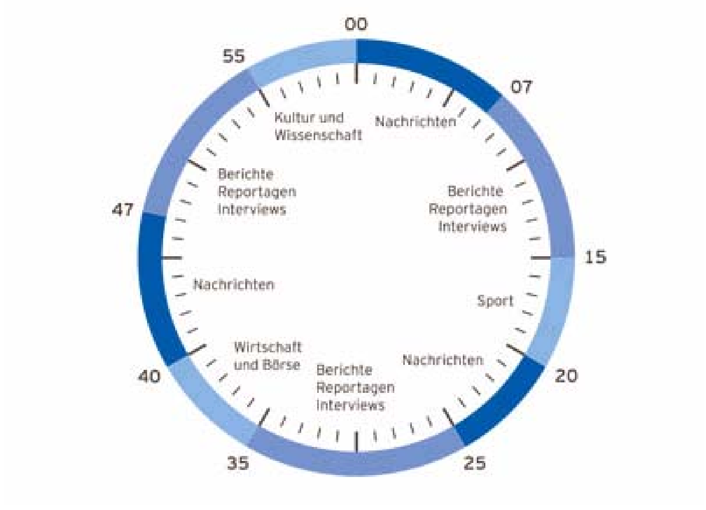


Figure 4 "The Radio Clock" – an abstract hourly schedule

The user will be enabled to fill some of these slots according to his/her own taste, while other slots will be fixed as they contain current information, advertisements or content important for the radio branding.

With a simplified mixing feature (e.g. with faders like on any traditional mixing device or equalizer, see below), users can define how much of which content source or type they want to use.

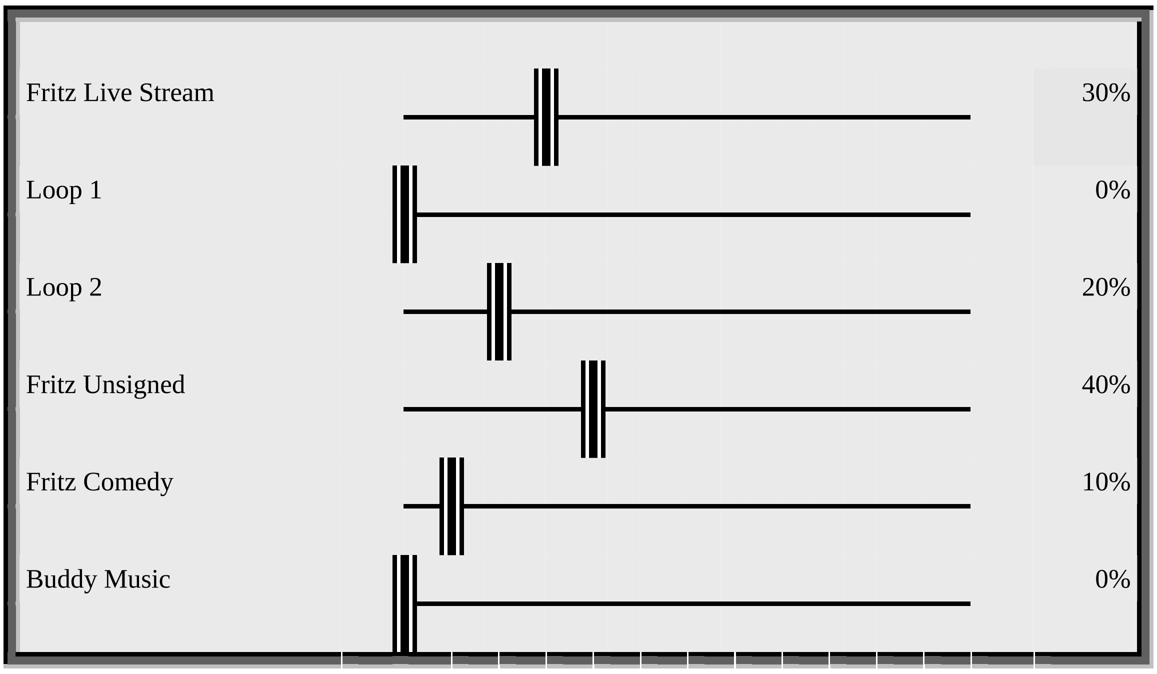


Figure 5 VITAL++ SoftRadio Mixer

Instead of tuning out (and not tuning back in – the worst case for a broadcaster), the user would keep the selected aspects and features of a programme, in other words stay with the programme as his favourite brand, and would be provided with different content during the time slots in-between semi-automatically.

Skipping and switching to other files than those automatically recommended, should always be possible.

As a second step and an alternative to semi-automatic recommendations according to a user-defined profile, users should be able to mix their own programme more exactly. Assuming, for instance, a university student (political sciences) as user, the scenario could be depicted like this:

1. The student travels one hour to university each day and one hour back (ignoring the fact that he would need a mobile VITAL++ capable device for a moment). He wants to customize a programme to listen to during this time.
2. The user chooses a linear radio programme as frame programme which basically provides current news, traffic, weather and advertisement. *Inforadio*, a news programme, would be a good choice for a student of political sciences. The parts of this frame programme will play live every full and half hour for approximately five minutes.
3. He is very much interested in the weekly documentary features on *Inforadio*, which are available as Podcast/on-demand stream. He picks three features – foreign politics update, science report and Berlin lifestyle guide. Each time when a new issue becomes available, it will be the first feature to be presented to him in the morning.
4. He likes to discover new music, so he adds “*Fritz unsigned*” offered by the youth radio station, which is broadcast every Sunday, so he will be offered this feature on Monday morning.
5. The other days he wants to listen to 15 minutes of user-generated video content taken from the online community platform “*www.meinfritz.de*”, highlights that have been picked by editors or rated very high by the visitors of the platform.
6. In the evening he wants to watch the regional info magazine Brandenburg aktuell, which would be streamed to him through this service.
7. The rest of the time, he simply wants to listen to music from his personal archive.

As mentioned above, with VITAL++ the content could be streamed (live programme or on demand) or users could share data with their buddies, e.g. listen to their music.

The sources to mix the individual programme from could be audio as well as video.

Through IMS any one item could be identified so as to prevent double copies of the same file (song/broadcast, ect.) which might not be recognized by the system and client as instances of the same file.

7.3 Scenario 3: Remote Rural Areas

In some remote rural areas, served by satellite connections or radio accesses, the use of P2P technologies can improve the way the operator serves multimedia on-demand content. Let's imagine a rural area where a number of users are connected to a broadband network using a number of satellite accesses. For VoD scenarios is probable that the same content is going to 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 a satellite access. This situation is depicted in the following drawing (see Figure 6).

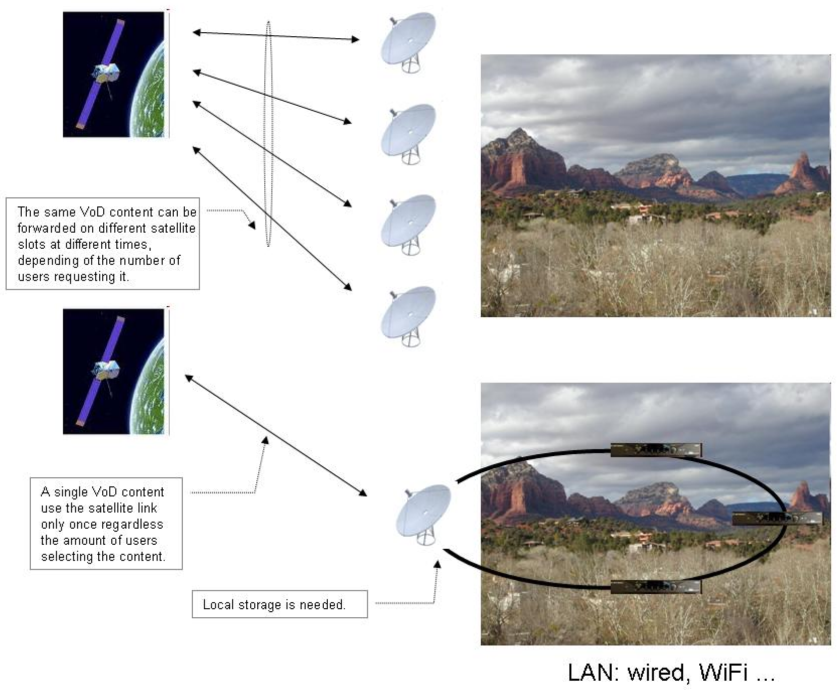

The same VoD content can be forwarded on different satellite slots at different times, depending of the number of users requesting it.
A single VoD content use the satellite link only once regardless the amount of users selecting the content.
Local storage is needed.
LAN: wired, WiFi ...

Figure 6 Remote Rural Area. VoD scenario

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 property of 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.

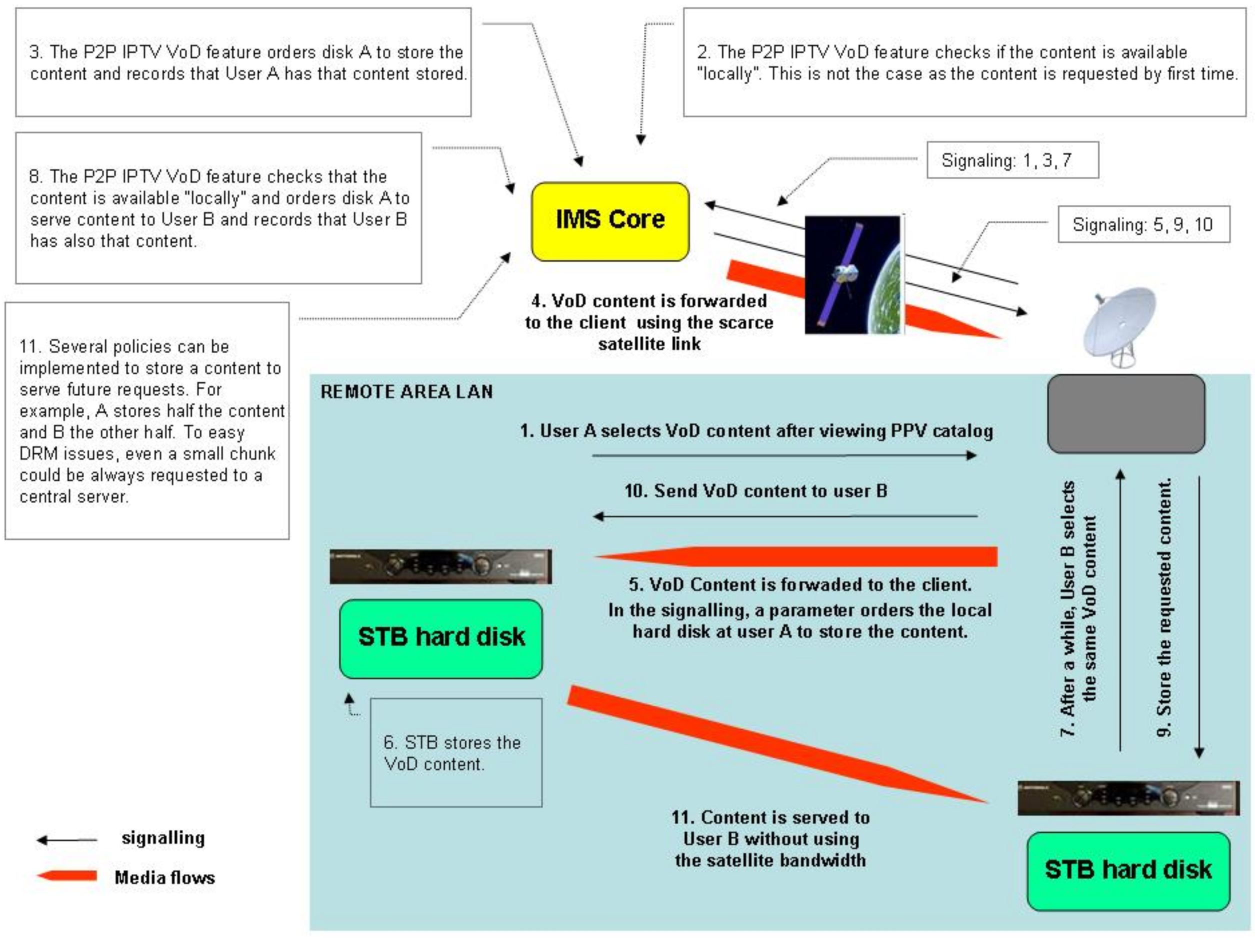

3. The P2P IPTV VoD feature orders disk A to store the content and records that User A has that content stored.
2. The P2P IPTV VoD feature checks if the content is available "locally". This is not the case as the content is requested by first time.
8. The P2P IPTV VoD feature checks that the content is available "locally" and orders disk A to serve content to User B and records that User B has also that content.
Signaling: 1, 3, 7
Signaling: 5, 9, 10
IMS Core
4. VoD content is forwarded to the client using the scarce satellite link
11. Several policies can be implemented to store a content to serve future requests. For example, A stores half the content and B the other half. To easy DRM issues, even a small chunk could be always requested to a central server.
REMOTE AREA LAN
1. User A selects VoD content after viewing PPV catalog
10. Send VoD content to user B
5. VoD Content is forwaded to the client.
In the signalling, a parameter orders the local hard disk at user A to store the content.
STB hard disk
6. STB stores the VoD content.
7. After a while, User B selects the same VoD content
9. Store the requested content.
11. Content is served to User B without using the satellite bandwidth
STB hard disk
signalling
Media flows

Figure 7 Remote Rural Area VoD scenario. Signalling and media flows.

8 Generic requirements/criteria regarding the clients

The following section describes the rationale for and process used to evaluate Peer-to-peer clients to determine their suitability for adaptation to meet the requirements of the VITAL++ project.

8.1 Reason for Evaluation

The VITAL++ project is investigating the merging of Peer-to-Peer technologies with IMS infrastructure. The resulting platform should have the scalability and fault-tolerant features that come with a decentralised P2P architecture while benefiting from IMS features such as strong authentication, encryption, auditing and accounting which are necessary for commercial deployment.

However, the integration of two technologies with very different concerns and motivations is not without its problems. IMS is realised through a collection of well specified logical nodes with defined interfaces and interactions required to achieve basic and advanced network functionality such as subscription, registration, session control, roaming and access network mediation. These nodes are centralised in a client-server architecture as, historically, they have been owned and operated in this fashion by network operators. A P2P-IMS involves the distribution of some of these nodes towards the edge of the network. In a P2P-IMS content storage, media processing and transcoding are all likely to be delegated to peer applications running on various mobile devices or set-top boxes. These peer applications will necessarily support client functionality required for connection to standard IMS infrastructure together with their P2P functionality for:

- Content search
- Content retrieval and distribution using streaming
- Content mixing

Work Package 2 focuses on VITAL++ Architecture Requirements and Specification. As part of this work package, Task 2.1 involves evaluating P2P clients and potentially libraries to determine suitability for use in achieving VITAL++ objectives.

From the Technical Annex:

"The main goal of this Task is to evaluate the multitude of existing Peer-to-Peer Software Clients with special focus on their feature set. This evaluation starts with an analysis of the usage scenarios and an examination of the client penetration over Europe. The most popular clients are investigated and their feature set compared amongst themselves. Special focus is set on the ability to

enhance/reuse the client software within IMS networks, the description of the extensibility of the Client interfaces in order to communicate within the IMS network surrounding is an integral part of the accompanying deadapted in the course of WP3."

As the VITAL++ platform will primarily address concerns surrounding content distribution this implies baseline functionality within the P2P client. Therefore we have described a set of evaluation criteria that permit us to decide upon one or more P2P clients and libraries, which are optimal for use.

8.2 Basis for Evaluation

The identified Evaluation Criteria are listed below.

Portability

The ability of the P2P client or library to be deployed on multiple operating systems and device types, thus ensuring that the VITAL++ platform can reach the maximum user community.

Concerns include:

- Implementation Language
- Operating System support
- Device support

Code Accessibility

The accessibility of the code for inclusion within and modification by the VITAL++ project. Issues that affect this include whether the code for the P2P client or library implementation is available to the general public or the consortium only. The license under which the code is distributed is also a factor as some licenses are highly restrictive and/or create compatibility issues with other 3rd party components.

Concerns include:

- Openness of Source Code
- License Model Used (e.g. BSD, GPL, APL etc.)

Scalability and Fault Tolerance

An assessment of the scalability features and fault tolerant features of the client's underlying P2P overlay. Any other issues of features of the client that will affect scalability of fault tolerance may also be considered.

Concerns include:

- P2P architecture including overlay topology
- IP version

- Detection and adaptation to changing network conditions.

Suitability for modification

Analysis of the client to determine whether it's a suitable starting point to realise the proposed VITAL++ scenarios. In particular, the structure of the code should be briefly examined to see that the design is intelligible and logical. A modular design that promotes extension is best.

Concerns include:

- Modularity
- Overlay abstraction/API
- Documentation of architecture and code.

SIP & IMS Suitability

SIP in general and the IMS architecture in particular require and suggest mechanisms for authentication, auditing and management. Here we consider the suitability of clients for integration within an IMS environment based on their existing design and capabilities. Any issues that could be detrimental to their adaptability for P2P-IMS scenarios, will also be considered. We assume client-side IMS functionality equivalent to the Rich Communications Suite initiative requiring registration, voice and video session control, media-sharing, contact management and presence.

Concerns include:

- AAA
- Other security concerns including Message Encryption and tamper-proofing
- Billing
- Manageability

Scenario Specific Features

The VITAL++ project will investigate the challenges of P2P IMS using a small number of appropriate service scenarios exercising functionality for distribution and consumption of content in various forms. Content here will range from basic “flat” files to live, time and sequence sensitive media streams, which are broadcast by and received by multiple parties. Difference scenarios involving different content types and access/distribution paradigms may require specific features from a client or library implementation that are not covered by other evaluation criteria.

Concerns include:

- Use of DHT;
- Support for isochronous media;

- Digital Rights Management (DRM) integration;
- Configurable content replication mechanisms;
- Anonymity of peers
- Rewards mechanism for good network etiquette and contribution;

8.3 Criteria and requirements for IMS operation

In the scope of this project, IMS functionalities shall be exploited to enrich the users experience at the VITAL++ hybrid client. Therefore, this client needs to be able to use the related IMS features transparently. IMS communication mainly means to fully support the Session Initiation Protocol (SIP) in version 2[22]. Additionally, the 3gpp has specified the exact behaviour of a client and its use of SIP messages, methods, transactions and header fields in a comprehensive technical specification[23]. In the following, the single IMS features required by the hybrid VITAL++ client are discussed.

Open API: As many features in the hybrid VITAL++ client require the assistance of central IMS functions, it is necessary to be able to influence the behaviour of the SIP/IMS stack in terms of transaction and especially header fields. E.g. it is necessary to transport DRM licenses, asymmetric encryption keys, etc. in the body of a SIP message.

Therefore, also the use of SIP over TCP is recommended, in order to avoid message size constrains when using UDP (a SIP message must fit into a single UDP packet, which is not bigger than the PMTU between the peers) and transaction timeout race conditions during cascaded P2PSIP operations, as P2PSIP is very likely to appear as a P2P protocol.

Also, the SIP state machine needs to be capable of being influenced in order to have the freeness to implement new transaction flows. At the moment it not clear, which of these features are really required, as many of them are related to question, how P2P communication and P2P/IMS-Core communication will be done. Nevertheless, for the design or choice of an IMS/SIP stack, this is an important point.

Recently, the JSR IMS API has been specified by the Java Community[24]. This API covers just a subset of the required features, which an extension in terms of P2P functionality might require. Therefore, the presence of this API might be desirable, but not necessary, as additional API elements must be created. Nevertheless, it is an interesting starting point, when a portable P2P extension is to be created.

[22] IETF RFC-3261

[23] 3GPP TS 24.229 V7.12.0 (06/2008)

[24] http://jcp.org/en/jsr/detail?id=281

Authentication: As the client acts as a normal IMS client towards the call state control functions, it must register with the IMS core before any further operations can be performed. Hence, the IMS authentication scheme must be supported.

Session Management: The IMS client must be capable of handling multimedia sessions. I.e. to initiate, modify and terminate them using IMS signalling.

Call referral: If the P2P management in the IMS application plane decides to rebuild or restructure the overlay, it will most likely do this by using the SIP REFER method in order to re-route media streams. Hence, support for this is needed in the IMS client.

Quality of Service: QoS is one of the most important features to be exploited. In IMS, QoS is realized by reserving bandwidth for a specific client by the call state control logic. In order to do this correctly, the client needs to supply its bandwidth requirements during SIP session establishment. For a VoIP call, this is done by the VoIP call handler module above the IMS/SIP stack. For a P2P media stream session, it must be possible to access the IMS/SIP stack in a way which allows modifying the related parameters.

Presence: Another goal within the project is to preserve the possibility to exchange private messages between users and to be informed when other users become available, regardless of the relation to those users. They can be friends, or peers with specific content, or peers for P2P request routing, etc. The IMS presence service allows to subscribe to events and to be notified when certain events occur. This generic mechanism will show itself useful when P2P mechanisms need to be supported.

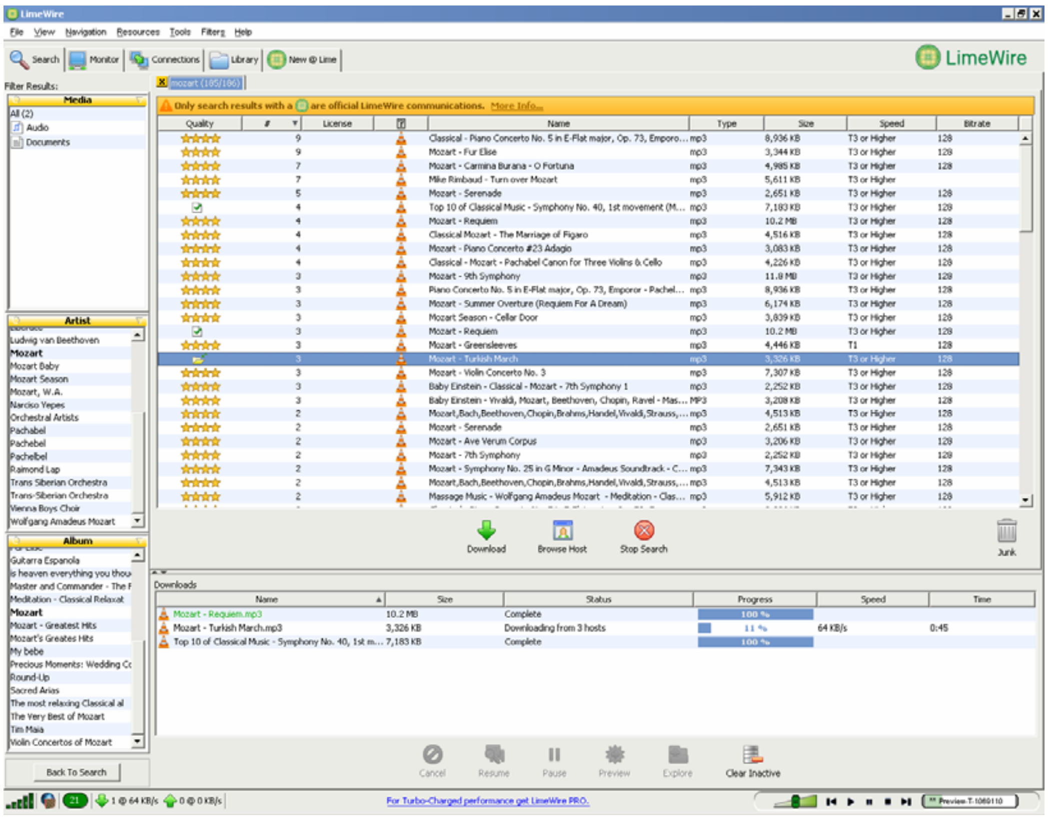
9 P2P Clients evaluation

This Section provides an overview of the existing most-popular P2P Clients, which are also evaluated (according to the specific usage scenario and following the criteria defined in the previous section), prior to the selection of the 3 most suitable ones for VITAL++.

9.1 P2P Clients for content distribution

9.1.1 Gnutella - Limewire

The LimeWire is a free open source client developed in java, which provides access mainly to audio files, which are located in the Gnutella network. A simplified example of how Gnutella works is a large circle were all node have the client software. When a client connect in the circle the node software must bootstrap and find at least one other node. Then the client will try to connect to the nodes, as well as nodes it receives from other clients, until it reaches a certain quota. Figure 8 and Figure 9 present the "Start-up screen" and the "Search Interface" respectively.



Only search results with a ● are official LimeWire communications. More Info...

Filter Results:

Media
- All (2)
- Audio
- Documents

Artist
- Ludwig van Beethoven
- **Mozart**
- Mozart Baby
- Mozart Season
- Mozart, W.A.
- Narciso Yepes
- Orchestral Artists
- Pachabel
- Pachebel
- Pachelbel
- Raimond Lap
- Trans Siberian Orchestra
- Trans-Siberian Orchestra
- Vienna Boys Choir
- Wolfgang Amadeus Mozart

Album
- Guitarra Espanola
- is heaven everything you thou
- Master and Commander - The F
- Meditation - Classical Relaxat
- **Mozart**
- Mozart - Greatest Hits
- Mozart's Greates Hits
- My bebe
- Precious Moments: Wedding Co
- Round-Up
- Sacred Arias
- The most relaxing Classical al
- The Very Best of Mozart
- Tim Maia
- Violin Concertos of Mozart

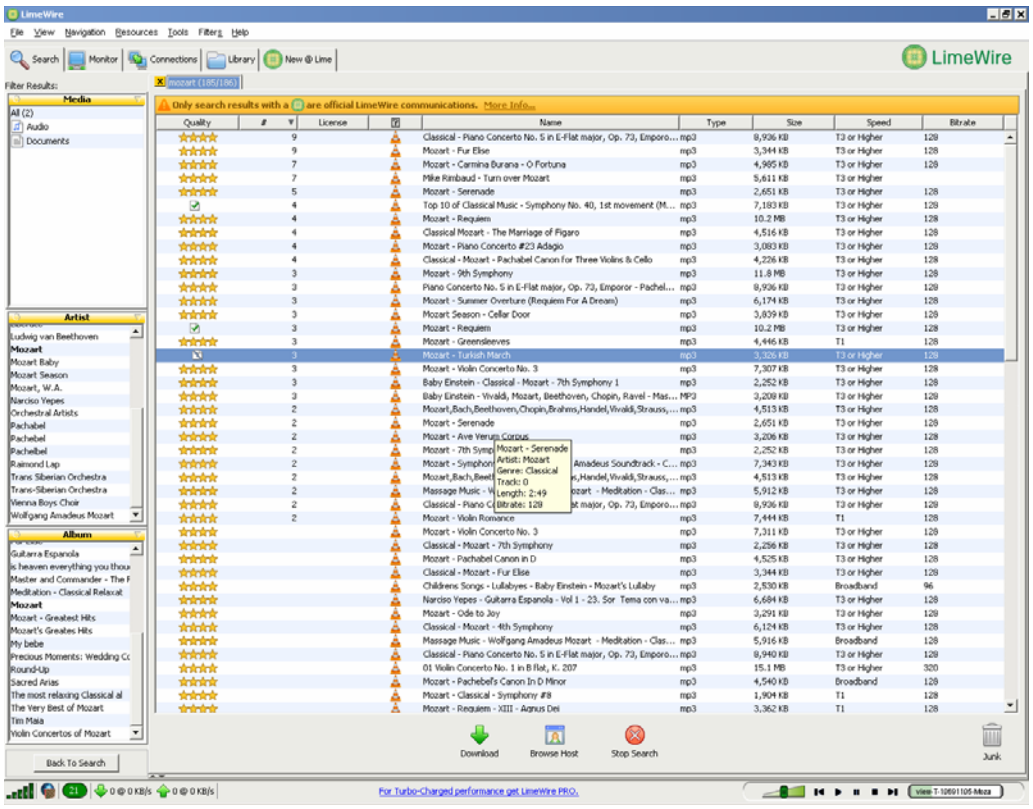
Quality	#	License		Name	Type	Size	Speed	Bitrate
★★★★★	9		▲	Classical - Piano Concerto No. 5 in E-Flat major, Op. 73, Emporo...	mp3	8,936 KB	T3 or Higher	128
★★★★★	9		▲	Mozart - Fur Elise	mp3	3,344 KB	T3 or Higher	128
★★★★★	7		▲	Mozart - Carmina Burana - O Fortuna	mp3	4,985 KB	T3 or Higher	128
★★★★★	7		▲	Mike Rimbaud - Turn over Mozart	mp3	5,611 KB	T3 or Higher	
★★★★★	5		▲	Mozart - Serenade	mp3	2,651 KB	T3 or Higher	128
✓	4		▲	Top 10 of Classical Music - Symphony No. 40, 1st movement (M...	mp3	7,183 KB	T3 or Higher	128
★★★★★	4		▲	Mozart - Requiem	mp3	10.2 MB	T3 or Higher	128
★★★★★	4		▲	Classical Mozart - The Marriage of Figaro	mp3	4,516 KB	T3 or Higher	128
★★★★★	4		▲	Mozart - Piano Concerto #23 Adagio	mp3	3,083 KB	T3 or Higher	128
★★★★★	4		▲	Classical - Mozart - Pachabel Canon for Three Violins & Cello	mp3	4,226 KB	T3 or Higher	128
★★★★★	3		▲	Mozart - 9th Symphony	mp3	11.8 MB	T3 or Higher	128
★★★★★	3		▲	Piano Concerto No. 5 in E-Flat major, Op. 73, Emporor - Pachel...	mp3	8,936 KB	T3 or Higher	128
★★★★★	3		▲	Mozart - Summer Overture (Requiem For A Dream)	mp3	6,174 KB	T3 or Higher	128
★★★★★	3		▲	Mozart Season - Cellar Door	mp3	3,839 KB	T3 or Higher	128
✓	3		▲	Mozart - Requiem	mp3	10.2 MB	T3 or Higher	128
★★★★★	3		▲	Mozart - Greensleeves	mp3	4,446 KB	T1	128
📂	3		▲	Mozart - Turkish March	mp3	3,326 KB	T3 or Higher	128
★★★★★	3		▲	Mozart - Violin Concerto No. 3	mp3	7,307 KB	T3 or Higher	128
★★★★★	3		▲	Baby Einstein - Classical - Mozart - 7th Symphony 1	mp3	2,252 KB	T3 or Higher	128
★★★★★	3		▲	Baby Einstein - Vivaldi, Mozart, Beethoven, Chopin, Ravel - Mas...	MP3	3,208 KB	T3 or Higher	128
★★★★★	2		▲	Mozart,Bach,Beethoven,Chopin,Brahms,Handel,Vivaldi,Strauss,...	mp3	4,513 KB	T3 or Higher	128
★★★★★	2		▲	Mozart - Serenade	mp3	2,651 KB	T3 or Higher	128
★★★★★	2		▲	Mozart - Ave Verum Corpus	mp3	3,206 KB	T3 or Higher	128
★★★★★	2		▲	Mozart - 7th Symphony	mp3	2,252 KB	T3 or Higher	128
★★★★★	2		▲	Mozart - Symphony No. 25 in G Minor - Amadeus Soundtrack - C...	mp3	7,343 KB	T3 or Higher	128
★★★★★	2		▲	Mozart,Bach,Beethoven,Chopin,Brahms,Handel,Vivaldi,Strauss,...	mp3	4,513 KB	T3 or Higher	128
★★★★★	2		▲	Massage Music - Wolfgang Amadeus Mozart - Meditation - Clas...	mp3	5,912 KB	T3 or Higher	128

Downloads

Name	Size	Status	Progress	Speed	Time
Mozart - Requiem.mp3	10.2 MB	Complete	100 %		
Mozart - Turkish March.mp3	3,326 KB	Downloading from 3 hosts	11 %	64 KB/s	0:45
Top 10 of Classical Music - Symphony No. 40, 1st m...	7,183 KB	Complete	100 %		

For Turbo-Charged performance get LimeWire PRO.

Figure 8 Limewire Start-up screen



Only search results with a ● are official LimeWire communications. More Info...

Quality	#	License		Name	Type	Size	Speed	Bitrate
★★★★★	9			Classical - Piano Concerto No. 5 in E-Flat major, Op. 73, Emporo...	mp3	8,936 KB	T3 or Higher	128
★★★★★	9			Mozart - Fur Elise	mp3	3,344 KB	T3 or Higher	128
★★★★★	7			Mozart - Carmina Burana - O Fortuna	mp3	4,985 KB	T3 or Higher	128
★★★★★	7			Mike Rimbaud - Turn over Mozart	mp3	5,611 KB	T3 or Higher	
★★★★★	5			Mozart - Serenade	mp3	2,651 KB	T3 or Higher	128
✓	4			Top 10 of Classical Music - Symphony No. 40, 1st movement (M...	mp3	7,183 KB	T3 or Higher	128
★★★★★	4			Mozart - Requiem	mp3	10.2 MB	T3 or Higher	128
★★★★★	4			Classical Mozart - The Marriage of Figaro	mp3	4,516 KB	T3 or Higher	128
★★★★★	4			Mozart - Piano Concerto #23 Adagio	mp3	3,083 KB	T3 or Higher	128
★★★★★	4			Classical - Mozart - Pachabel Canon for Three Violins & Cello	mp3	4,226 KB	T3 or Higher	128
★★★★★	3			Mozart - 9th Symphony	mp3	11.8 MB	T3 or Higher	128
★★★★★	3			Piano Concerto No. 5 in E-Flat major, Op. 73, Emporor - Pachel...	mp3	8,936 KB	T3 or Higher	128
★★★★★	3			Mozart - Summer Overture (Requiem For A Dream)	mp3	6,174 KB	T3 or Higher	128
★★★★★	3			Mozart Season - Cellar Door	mp3	3,839 KB	T3 or Higher	128
✓	3			Mozart - Requiem	mp3	10.2 MB	T3 or Higher	128
★★★★★	3			Mozart - Greensleeves	mp3	4,446 KB	T1	128
☒	3			Mozart - Turkish March	mp3	3,326 KB	T3 or Higher	128
★★★★★	3			Mozart - Violin Concerto No. 3	mp3	7,307 KB	T3 or Higher	128
★★★★★	3			Baby Einstein - Classical - Mozart - 7th Symphony 1	mp3	2,252 KB	T3 or Higher	128
★★★★★	3			Baby Einstein - Vivaldi, Mozart, Beethoven, Chopin, Ravel - Mas...	MP3	3,208 KB	T3 or Higher	128
★★★★★	2			Mozart,Bach,Beethoven,Chopin,Brahms,Handel,Vivaldi,Strauss,...	mp3	4,513 KB	T3 or Higher	128
★★★★★	2			Mozart - Serenade	mp3	2,651 KB	T3 or Higher	128
★★★★★	2			Mozart - Ave Verum Corpus	mp3	3,206 KB	T3 or Higher	128
★★★★★	2			Mozart - 7th Symp	mp3	2,252 KB	T3 or Higher	128
★★★★★	2			Mozart - Symphon Amadeus Soundtrack - C...	mp3	7,343 KB	T3 or Higher	128
★★★★★	2			Mozart,Bach,Beet s,Handel,Vivaldi,Strauss,...	mp3	4,513 KB	T3 or Higher	128
★★★★★	2			Massage Music - W ozart - Meditation - Clas...	mp3	5,912 KB	T3 or Higher	128
★★★★★	2			Classical - Piano C t major, Op. 73, Emporo...	mp3	8,936 KB	T3 or Higher	128
★★★★★	2			Mozart - Violin Romance	mp3	7,444 KB	T1	128
★★★★★				Mozart - Violin Concerto No. 3	mp3	7,311 KB	T3 or Higher	128
★★★★★				Classical - Mozart - 7th Symphony	mp3	2,256 KB	T3 or Higher	128
★★★★★				Mozart - Pachabel Canon in D	mp3	4,525 KB	T3 or Higher	128
★★★★★				Classical - Mozart - Fur Elise	mp3	3,344 KB	T3 or Higher	128
★★★★★				Childrens Songs - Lullabyes - Baby Einstein - Mozart's Lullaby	mp3	2,530 KB	Broadband	96
★★★★★				Narciso Yepes - Guitarra Espanola - Vol 1 - 23. Sor Tema con va...	mp3	6,684 KB	T3 or Higher	128
★★★★★				Mozart - Ode to Joy	mp3	3,291 KB	T3 or Higher	128
★★★★★				Classical - Mozart - 4th Symphony	mp3	6,124 KB	T3 or Higher	128
★★★★★				Massage Music - Wolfgang Amadeus Mozart - Meditation - Clas...	mp3	5,916 KB	Broadband	128
★★★★★				Classical - Piano Concerto No. 5 in E-Flat major, Op. 73, Emporo...	mp3	8,940 KB	T3 or Higher	128
★★★★★				01 Violin Concerto No. 1 in B flat, K. 207	mp3	15.1 MB	T3 or Higher	320
★★★★★				Mozart - Pachebel's Canon In D Minor	mp3	4,540 KB	Broadband	128
★★★★★				Mozart - Classical - Symphony #8	mp3	1,904 KB	T1	128
★★★★★				Mozart - Requiem - XIII - Agnus Dei	mp3	3,362 KB	T1	128

Mozart - Serenade
Artist: Mozart
Genre: Classical
Track: 0
Length: 2:49
Bitrate: 128

Figure 9 LimeWire Search Interface

9.1.1.1 Categories

This section evaluates LimeWire software within the categories defined in section 8.

Portability

The LimeWire is a front end client for the Gnutella/G2 network, written in java provides a reliable client supporting various Operating systems such us Windows Mac OS and the ubuntu-debian Linux distributions. LimeWire is supported by the Gnutella forum, currently they are working in implementing a portable client for mobile devices and a client that support the bit torrent protocol.

Code Accessibility

LimeWire is a free peer-to-peer file sharing (P2P) client written in the Java platform, it use the Gnutella network. The Basic edition of the LimeWire is free of charge and compared with the pro edition does not provide extensive tech support and optimum search results. LimeWire basic edition is open source and it is developed online providing resent revision via the subversion technology (CVS) and it can be downloaded via the wiki LimeWire page. The LimeWire is under the GNU General Public License (GPL).

Scalability and Fault Tolerance

Lime Wire supports Gnutella's open-protocol. The gnutella protocol is a highly scalable protocol as described by the Gnutella Developer forum[25]: *Gnutella protocol distinction is its peer-to-peer, decentralized model. In this model, every client is a server, and vice versa. These so-called Gnutella servents perform tasks normally associated with both clients and servers. They provide client-side interfaces through which users can issue queries and view search results, while at the same time they also accept queries from other servents, check for matches against their local data set, and respond with applicable results. Due to its distributed nature, a network of servents that implements the Gnutella protocol is highly fault-tolerant, as operation of the network will not be interrupted if a subset of servents goes offline.*

LimeWire supports IPv6

Suitability for modification

A large community supports LimeWire and the Gnutella network and it can be a good and suitable P2P client for the implementation of P2P technology in IMS network which is suggested by VITAL++. The help and support that Gnutella Developer Forum can provide a valuable assistant in the deployment of an IMS P2P LimeWire like client for the VITAL++ testbed.

The source code is available in the following are for download:

Online https://www.limewire.org/fisheye/viewrep/limecvs/

Subversion cvs -d:pserver:guest@cvs.limewire.org:/cvs login

The code appears to be clearly laid out with a well structure and well documented .

SIP & IMS Suitability

- *AAA*: The LimeWire support Host that are located via the IP address and there is a module that Hosts can communicate but not AAA modules are available by the Gnutella Protocol.
- *Other security concerns including Message Encryption and tamper-proofing:* LimeWire use Ultrapeer, Mojito DHT and TLS. LimeWire uses the Mojito Distributed Hash Table (DHT) to find more sources for downloads. Also, it support TLS

[25] http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

- Billing: This is not a feature of the Gnutella protocol..
- Manageability: This is not a feature of the Gnutella protocol.

Scenario Specific Features

- *Use of DHT*: DHT stands for Distributed Hash Table. LimeWire use Mojito Distributed Hash Table To explain how the DHT works we take O(log N) hops, where N is the network size. Assuming each hop takes t milliseconds, then a search should take O(log N)*t (or O(log N)) milliseconds. Although the search delays grow with the network size, the growth is really slow. In addition, in theory, without increasing the network bandwidth cost much, the search time could be improved to C*t with proper optimizations, where C is a constant. This means, although the number of hops grow with the network size, the search delays do not grow. LimeWire use a modified kademlia implementation named Mojito Distributed Hash.

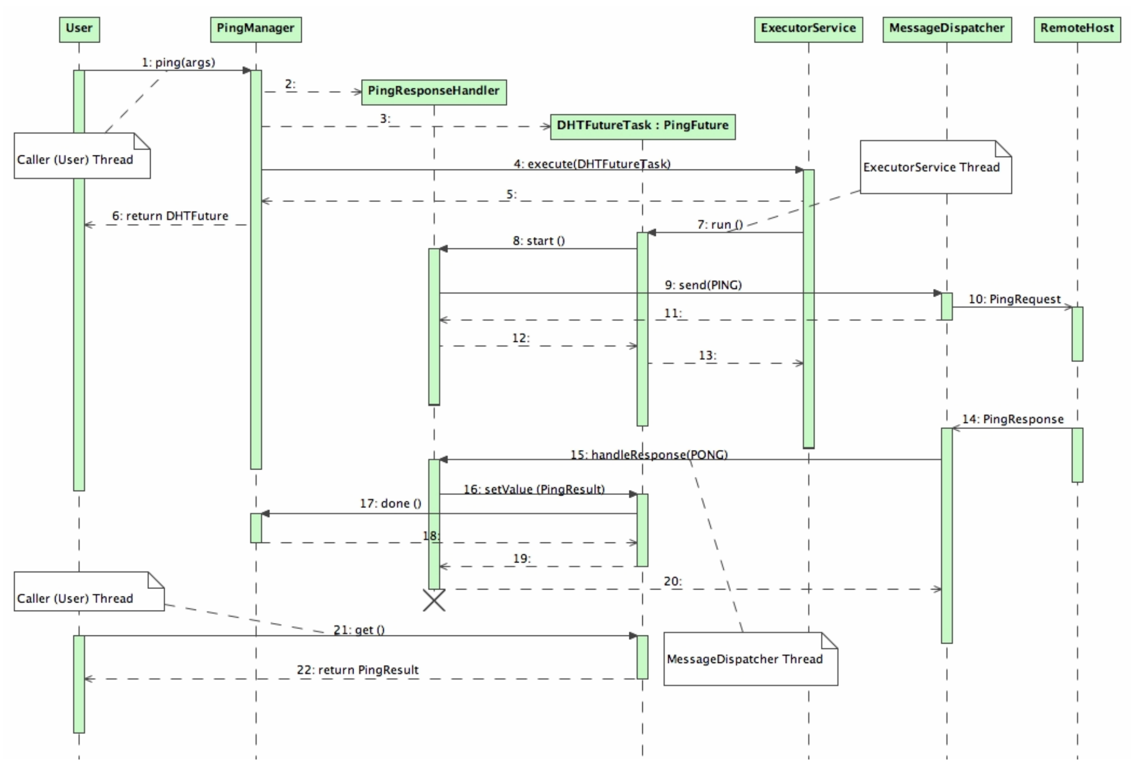


Figure 10 Mojito Distributed Hash table Ping Sequence

- *Support for isochronous media*: This is a feature not supported by the LimeWire client.

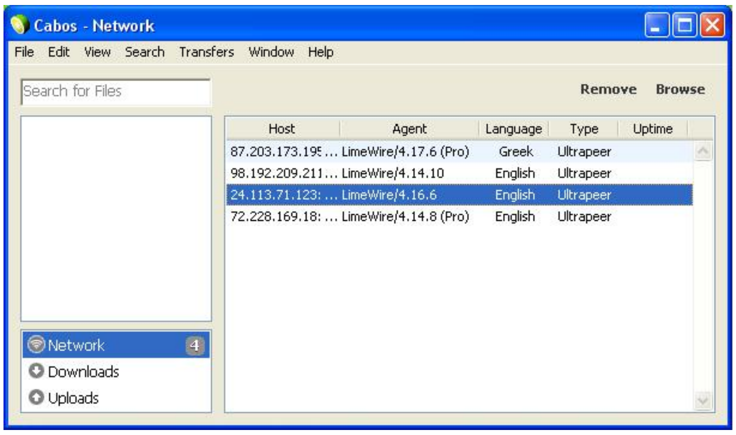
- *Digital Rights Management (DRM) integration: Lime Wire is introducing a filtering system to encourage safer, more responsible file sharing. Copyright owners interested in blocking their files from being downloaded, uploaded and shared are invited to register with LimeWire community. LimeWire users can learn more about responsible file sharing in Copyright Information that is provided online.*
- *Configurable content replication mechanisms:* LimeWire is a gateway to the Gnutella network. Like other Gnutella-compatible software programs, LimeWire allows individual computers to connect and to search other computers on the Gnutella Network. When a file is been found it can be downloaded directly from the computer that holds it.

9.1.1.2 Concluding remarks

LimeWire is a fast, user friendly file sharing program that provides no anonymity to the user. LimeWire is opensource under GNU/GPL providing a big community to the developers. LimeWire is developed in java and a large developer's community supports it. Limewire is a file sharing application that does not support the ability to stream video or audio. Finally Lime Wire is introducing a filtering system to encourage safer, more responsible file sharing providing a notion of DRM.

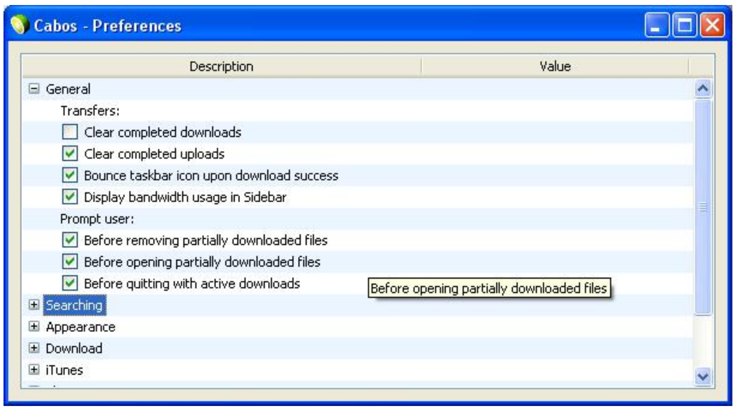
9.1.2 Cabos

Cabos is an open source Gnutella file sharing program based on LimeWire and Acquisition. Cabos provides a user friendly environment with the ability to transfer via firewall, proxy's etc. Figure 11, Figure 12 and Figure 13 present the Cabos' network selection, the preferences menu and the download/file-sharing section respectively.



Host	Agent	Language	Type	Uptime
87.203.173.195…	LimeWire/4.17.6 (Pro)	Greek	Ultrapeer	
98.192.209.211…	LimeWire/4.14.10	English	Ultrapeer	
24.113.71.123: …	LimeWire/4.16.6	English	Ultrapeer	
72.228.169.18: …	LimeWire/4.14.8 (Pro)	English	Ultrapeer	

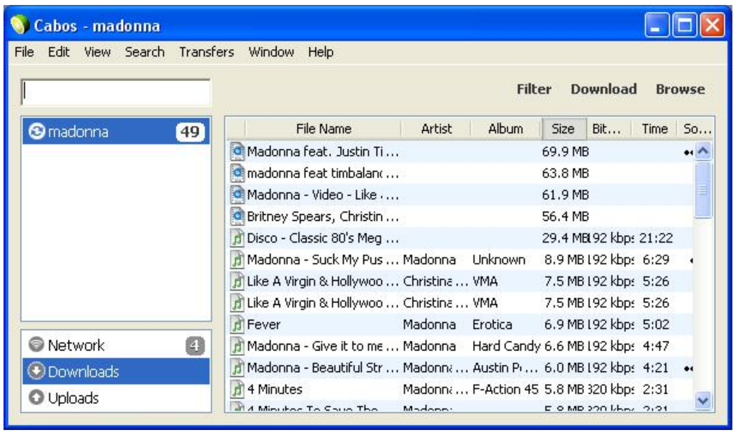
Figure 11 Cabos network selection



⊟ General
- Transfers:
  - ☐ Clear completed downloads
  - ☑ Clear completed uploads
  - ☑ Bounce taskbar icon upon download success
  - ☑ Display bandwidth usage in Sidebar
- Prompt user:
  - ☑ Before removing partially downloaded files
  - ☑ Before opening partially downloaded files
  - ☑ Before quitting with active downloads

⊞ Searching
⊞ Appearance
⊞ Download
⊞ iTunes

Figure 12 Cabos Preference



File Name	Artist	Album	Size	Bit...	Time	So...
Madonna feat. Justin Ti…			69.9 MB			
madonna feat timbalan…			63.8 MB			
Madonna - Video - Like …			61.9 MB			
Britney Spears, Christin…			56.4 MB			
Disco - Classic 80's Meg…			29.4 MB	192 kbps	21:22	
Madonna - Suck My Pus…	Madonna	Unknown	8.9 MB	192 kbps	6:29	
Like A Virgin & Hollywoo…	Christina…	VMA	7.5 MB	192 kbps	5:26	
Like A Virgin & Hollywoo…	Christina…	VMA	7.5 MB	192 kbps	5:26	
Fever	Madonna	Erotica	6.9 MB	192 kbps	5:02	
Madonna - Give it to me…	Madonna	Hard Candy	6.6 MB	192 kbps	4:47	
Madonna - Beautiful Str…	Madonna…	Austin P…	6.0 MB	192 kbps	4:21	
4 Minutes	Madonna…	F-Action 45	5.8 MB	320 kbps	2:31	

Figure 13 Cabos File sharing

9.1.2.1 Categories

This section evaluates CABOS software within the criteria defined in section 8.

Portability

The Cabos is a front-end client for the Gnutella/G2 network, Cabos is based on LimeWire and Acquisition. Cabos supports only Windows and Mac OS platform. The community of Cabos is based in Japan. Cabos use the Realbasic for its GUI, Realbasic is creating a cross platform for supporting mobile devises but until now Cabos network is not implementing any client that will support mobile devices.

Code Accessibility

Cabos is a free peer-to-peer file sharing (P2P) client the core application is written in java and the Gui in REALbasic. Cabos is under the GNU General Public License(GPL) the community is based in Japan and the source code can be found in the Japan source forge under the name CABOS.

Scalability and Fault Tolerance

Cabos supports Gnutella's open-protocol such us LimeWire but it lacks the chat and library features found in LimeWire. Cabos based can be highly scalable and resist in fault tolerance because is based in Gnutella protocol.

Suitability for modification

Cabos is based in japan and it has a medium size developer's community it is not supported by the Gnutella network. Cabos use REALbasic for the Gui which is not a popular language and there are not many applications created by REALbasic.

The source code is available in the Japan sourceforge:

Subversion http://svn.sourceforge.jp/cgi-bin/viewcvs.cgi/?root=cabos
The source code appears to be clearly laid out but it is poorly documented.

SIP & IMS Suitability

- *AAA*: The Cabos does not support any lever of AAA.
- *Other security concerns including Message Encryption and tamper-proofing:* Cabos is based in LimeWire it use also Ultrapeer, Mojito DHT or TLS.
- Billing: The Cabos does not support billing.
- Manageability: This is not a feature of the Gnutella protocol.

Scenario Specific Features

- *Use of DHT*: Same as Limewire
- Support for isochronous media: This is a feature not supported by the Cabos client
- *Digital Rights Management (DRM) integration*: Same as Limewire.

Configurable content replication mechanisms: Same as Limewire

9.1.2.2 Concluding remarks

Cabos seems unsuitable for VITAL++ project, mainly to its inability for supporting SIP and IMS functions/operations.

9.2 P2P Clients for VoD

9.2.1 Azureus (Vuze)

Azureus is a BitTorrent client written in Java and operating system independent. Vuze allows users to view, publish and share original DVD and HD quality video content. Content is presented through channels and categories containing TV shows, music videos, movies, video games and others. Additionally, if users prefer to publish their original content the just upload it to the Vuze search engine.

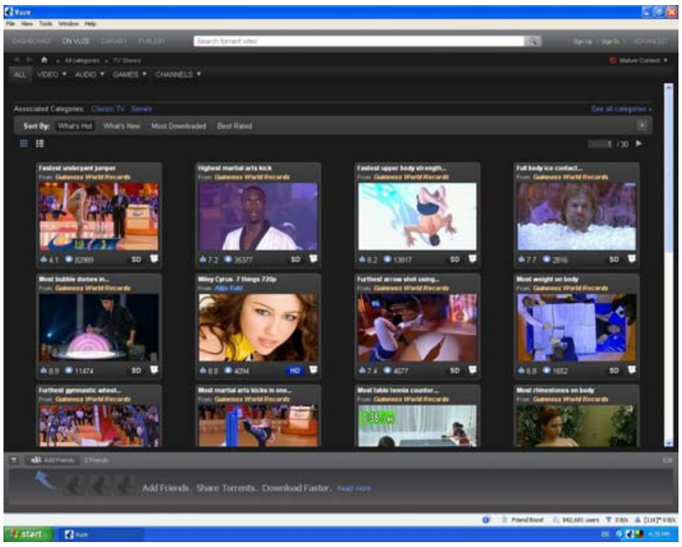


Figure 14 Video selection

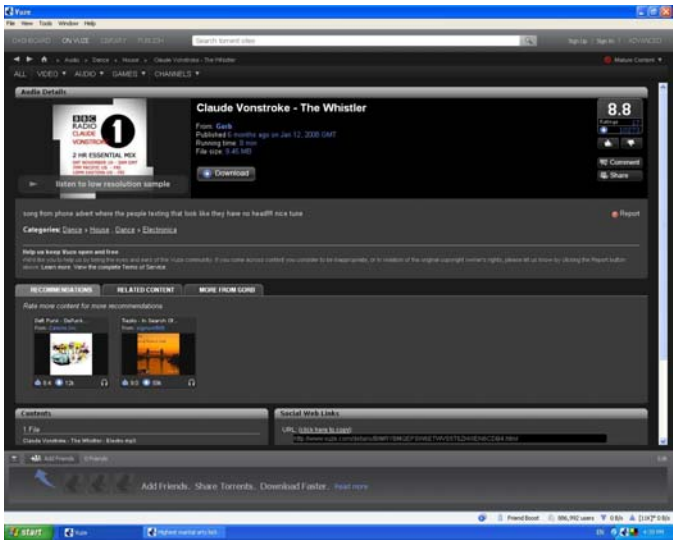


Figure 15 Download of a video

9.2.1.1 Categories

This section evaluates Vuze software within the following categories, these categories are defined in the section above in general terms.

Portability

Portability can be seen from different perspectives e.g. operating system, programming language, processor, devices adaptation, access to I/O devices and network resources. Vuze is mostly written in java programming language, which makes it portable across various platforms (Windows, Mac OS X, Unix/Linux). There is no version of Vuze confirmed to be installed on mobile phone or hand held devices. However since it is written in java it may be relatively portable to a mobile device supporting the J2ME Connected Device Configuration.

Code Accessibility

Vuze is commercial software, which uses the open source Azureus client engine. The original Azureus up till version 2.5.0.4 was available under the GNU General public License (GPL). After the version 3 distribution the licence is changed, while it still states that Azureus application is available under the GPL but after the completion of installation it requires the user to agree to the terms of "Vuze platform" which include restriction on use, reverse-engineering and sublicensing. The source code of Vuze software is available via sourceforge repository under the module named Azureus[26]3.

Scalability and Fault Tolerance

Vuze maintains its own DHT, which is incompatible with the official BitTorrent clients offered by BitTorrent, Inc. However for the compatibility purpose an implementation of the original BitTorrent DHT is available as plug-in.

A BitTorrent tracker is a server, which helps the communication between the peers using the BitTorrent protocol. It is a major critical point in the absence of the extensions to the original protocol as clients have to communicate with the tracker to initiate the download.

The concept of peer exchange has been introduced in Azureus 2.3.0.0. The peer exchange (PEX) help to reduce load on trackers and the distributed database. Peers only need them as an initial source of peers and then depend on the peers supporting PEX. However they report back to tracker as well but after max-interval thus reducing the load. This exchange is managed through the Azureus messaging protocol.

[26] Vuze Licence
http://www.azureuswiki.com/index.php/Azureus_2_/_3_and_Vuze#Is_the_source_for_Vuze_available.3F

Azureus supports IPv6 on Unix/Linux and Mac OS X, however IPV6 support on windows is not yet available[27].

Suitability for modification

Vuze could be a good and suitable starting point for a P2P implementation in IMS network which is suggested by VITAL++ as the Azureus engine is open source software distributed under GNU/GPL. Its structure is can be subdivided into:

- Azureus core + Standard 'SWT' UI: http://azureus.cvs.sourceforge.net/azureus/azureus2/
- Vuze UI: http://azureus.cvs.sourceforge.net/azureus/azureus3/
- Console UI: http://azureus.cvs.sourceforge.net/azureus/uis

The code appears to be clearly laid out but the client does not specify a platform API, as such. However, we feel that suitable abstractions could be implemented on top of the existing codebase.

SIP & IMS Suitability

- *AAA*: The Authentication model currently implemented by Vuze involves an authenticating server, rather than true distributed authentication models such as SPKI. A centralised authentication architecture is suitable for IMS integration. BitTorrent generates the tracker log that can be used as an audit trail. Tamper proofing mechanisms could be introduced if log files are to be used for non-repudiation.
- *Other security concerns including Message Encryption and tamper-proofing:* Torrent files and pieces are checksummed using the one-way Sha1 hash. However, the concept of using PKI for digitally signing content was not envisaged by the BitTorrent creators. BitTorrent encryption has primarily evolved out of a desire to obfuscate BitTorrent traffic, thus reducing ISP's ability to filter or throttle it. It is not intended to provide anonymity or confidentiality. VUZE supports the latest (early 2006) Protocol Encryption (PE) specification for BitTorrent header and message encryption
- Billing: This is not a feature of the BitTorrent protocol but it may be added to client in order to retrieve downloadable torrents.
- Manageability: This is not a feature of the BitTorrent protocol.

[27] http://www.sixxs.net/tools/tracker/clients

Scenario Specific Features

- *Use of DHT*: DHT stands for Distributed Hash Table. In Vuze the distributed database is based on a UDP based DHT. In fact Azureus uses a modified kademlia implementation. DHT based on the SHA-1 hash of the node's IP/Port combination and support 4 basic operation
    - Ping – to ensure up to date routing tables.
    - Lookup node – to find nodes that are near to the desired key in the key space
    - Get value – store a single value or a list ofvalues on these nodes
- *Support for isochronous media*: Technically it is possible to broadcast live streaming using the bit torrent protocol. Although this feature is not implemented in Vuze but this is demonstrated in swarm player that modifies the piece-picking and upload policies of bittorrent[28]. This gives us fair idea that we can incorporate live streaming using bit torrent.
- *Digital Rights Management (DRM) integration*: This aspect it very crucial in P2P networks, Vuze has a strong emphasis on community; users are able to create channels for content, rate and have conversation around the content. This is another strong feature of Vuze that allows a user to create a social network. Producers can upload their original works and can even charge a download or rental fee. User can not share DRM'd data or content that he has purchased or rented on Vuze[29].
- *Configurable content replication mechanisms:* The Vuze node distributing a data file treats the file as a number of **identically-sized** pieces, typically between 64 kB and 4 MB each. The peer creates a checksum for each piece, using the SHA1 hashing algorithm, and records it in the torrent file. Pieces with sizes greater than 512 kB will reduce the size of a torrent file for a very large payload, but is claimed to reduce the efficiency of the protocol. When another peer later receives a particular piece, the checksum of the piece is compared to the recorded checksum to test that the piece is error-free.Vuze distributes content as torrent files which contain the address of the tracker server and the hash Identifier for the torrent. The group of nodes that are active on a single torrent are known as a swarm. A tracker server contains the information needed for peers to connect to other peers. Trackers coordinate the BitTorrent clients, and also keep track of statistics and verification information for each torrent. Azureus contains its own built-in tracker, but there are a variety of other tracker software packages in use.

[28] Tribler, http://www.tribler.org/LiveStreamingBeta

[29] Vuze DRM http://faq.vuze.com/?View=entry&EntryID=231

BitTorrent nodes fall into 2 categories:

1. Peers
2. Seeds.

Seed nodes have 100% of the torrent and upload to other peers. In BitTorrent the term seed is also used as a verb, as in "I'm going to be seeding this file overnight", meaning leave open and available for other users to download. Usually, a person with the whole file is a 'seed', while someone with a partial amount is a peer. BitTorrent will try to balance the amount of seeds it connects to in a swarm with the number of peers.

Peers are nodes, which upload as well as download data for a torrent. i.e. they're sharing data as well as receiving it.

- Anonymity of peers: There is no built in mechanism provided by Azureus to keep peers anonymous however this functionality can be included with the help of a NodeZilla plug-in.
- *Rewards mechanism* for good network etiquette and contribution: BitTorrent also rewards peers, which contribute resources to the network through a process known as choking. Choking is a signal that a peer is not intending to send you any data, until you are unchoked. This could be because the peer is not ready, or willing to fulfill your requests. When you are connected to a peer, the connection contains information on 2 situations,
    1. **choked** or **unchoked**,
    2. **interested** or **not interested**.

Interested means, that peer maintain data that you do not have, and wish to acquire. New connections to peers always begin as choked and not interested. Peers will unchoke connected peers who upload fast but are not interested. If the fast uploading peers subsequently become interested, then the worst up-loader gets choked. If you are interested in what the peer has, then that peer's client will calculate whether to send you the data, using information such as how fast you are uploading to other peers. When you are connected to many peers, it is statistically improbable that you will receive data from all of them at once. Sometimes data is sent to you from a particular peer, sometimes that peer sends to another peer instead, who may then upload to you.

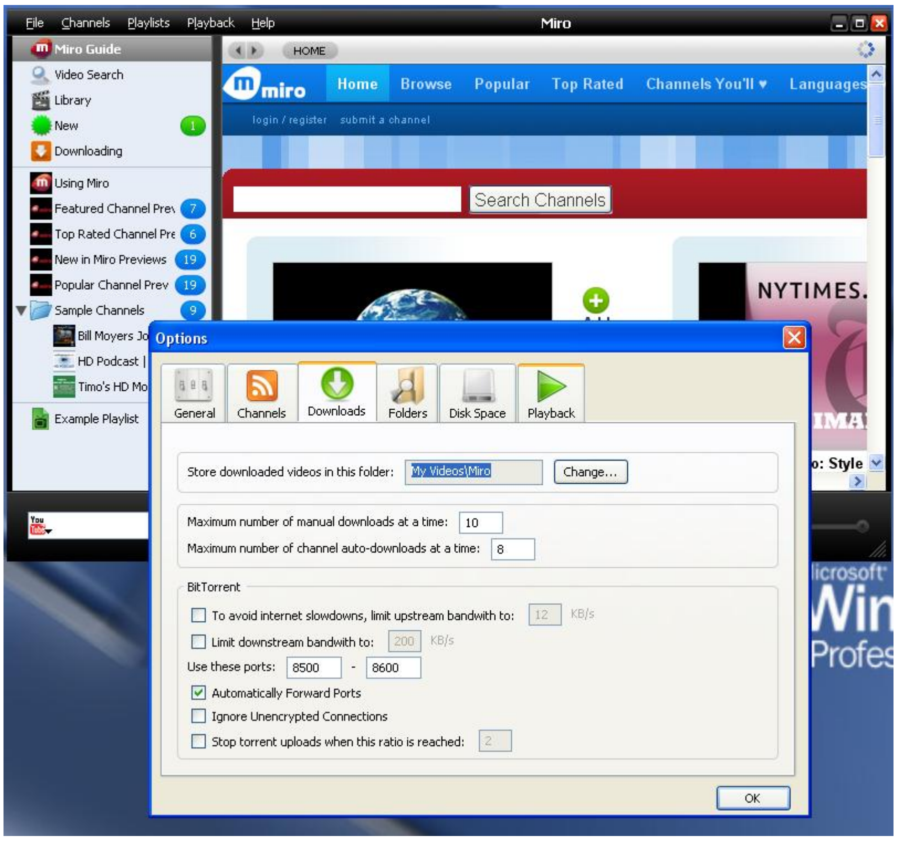
It is not possible for every peer to share data with every other connected peer at the same time. The TCP-Protocol used in BitTorrent to connect to other peers gets easily congested, and performs badly when sharing data over many connections at once. So choking is used to limit that congestion, and to help make sharing faster. Choking is also used to make sharing fairer to all, by ensuring that peers who upload more data faster to others get more data uploaded to them.

9.2.1.2 Concluding remarks

The Vuze P2P client is based on the open source Azureus Java client making it highly portable. It uses an up-to-date implementation of the BitTorrent protocol with support for message and header encryption. It uses Peer Exchange Protocol to improve fault tolerance and scalability. This is configurable permitting centralised tracking of streams where the publisher wishes. Vuze uses a central server for authentication. This has drawbacks in terms of fault tolerance but arguably eases IMS integration. Node IP addresses are not anomymised which lends itself to the content geoblocking scenario. Vuze is not designed to support multimedia conferencing scenarios nor does it support live streaming. The Azureus client has a plugin-architecture with free plugins available for SMS notification, RSS channel creation and download etc. Allowing users to create content channels is appropriate for the Personal Radio scenario. Azureus uses a "virtual positioning system" called Vivaldi to optimise communications between peers based on statistics such as network ping times. This may be useful in realising the remote rural access scenario. It appears to be highly suitable for the development of file distribution and Video-on-Demand applications.

9.2.2 Miro

Miro formerly known as ‘Democracy Player’ is an opens source platform created by a non-profit organization named participatory culture foundation. Miro is a desktop video application designed to make mass media more open and accessible for everyone. Miro is based on python with platform specific frontends for Windows, OS X, and GTK/X11. Miro aims to make online video “as easy as watching TV”; while at the same time ensuring that the new medium remains accessible to everyone, through its support for open standards.



Store downloaded videos in this folder: My Videos\Miro

Maximum number of manual downloads at a time: 10

Maximum number of channel auto-downloads at a time: 8

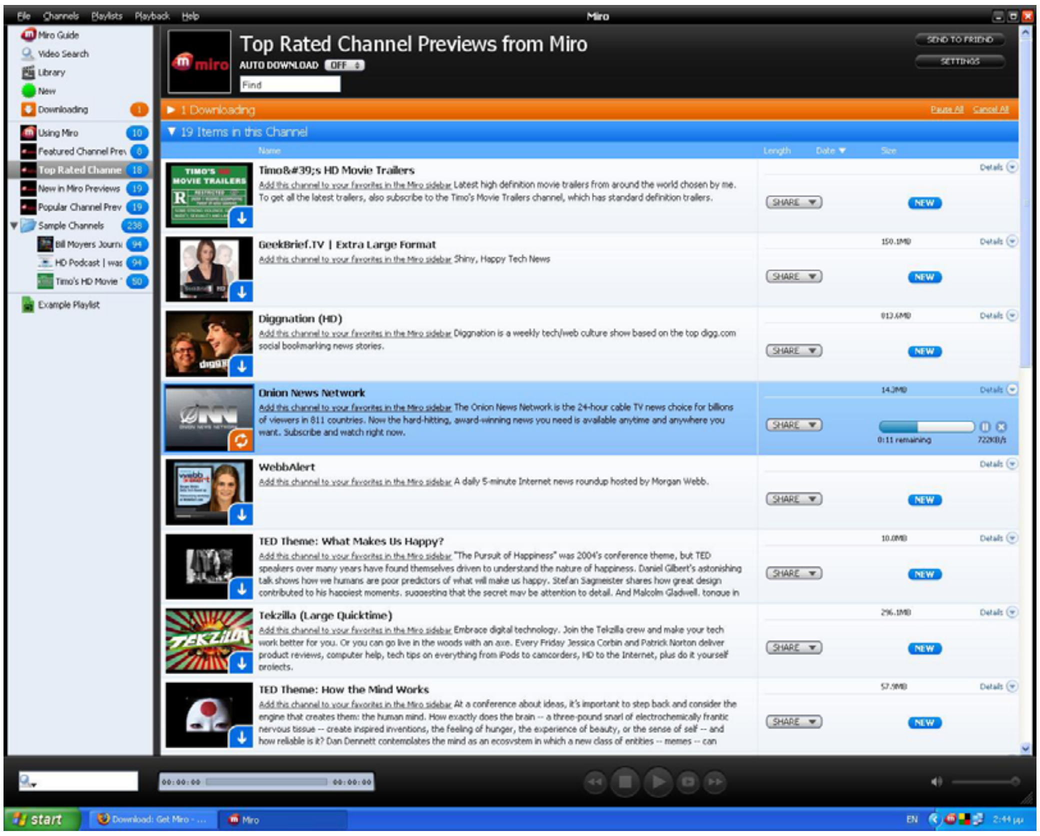
BitTorrent

- [ ] To avoid internet slowdowns, limit upstream bandwith to: 12 KB/s
- [ ] Limit downstream bandwith to: 200 KB/s

Use these ports: 8500 - 8600

- [✓] Automatically Forward Ports
- [ ] Ignore Unencrypted Connections
- [ ] Stop torrent uploads when this ratio is reached: 2

Figure 16 Miro preferences



Top Rated Channel Previews from Miro

1 Downloading

19 Items in this Channel

Timo&#39;s HD Movie Trailers
Add this channel to your favorites in the Miro sidebar Latest high definition movie trailers from around the world chosen by me. To get all the latest trailers, also subscribe to the Timo's Movie Trailers channel, which has standard definition trailers.

GeekBrief.TV | Extra Large Format 150.1MB
Add this channel to your favorites in the Miro sidebar Shiny, Happy Tech News

Diggnation (HD) 813.6MB
Add this channel to your favorites in the Miro sidebar Diggnation is a weekly tech/web culture show based on the top digg.com social bookmarking news stories.

Onion News Network 14.3MB
Add this channel to your favorites in the Miro sidebar The Onion News Network is the 24-hour cable TV news choice for billions of viewers in 811 countries. Now the hard-hitting, award-winning news you need is available anytime and anywhere you want. Subscribe and watch right now.
0:11 remaining 722KB/s

WebbAlert
Add this channel to your favorites in the Miro sidebar A daily 5-minute Internet news roundup hosted by Morgan Webb.

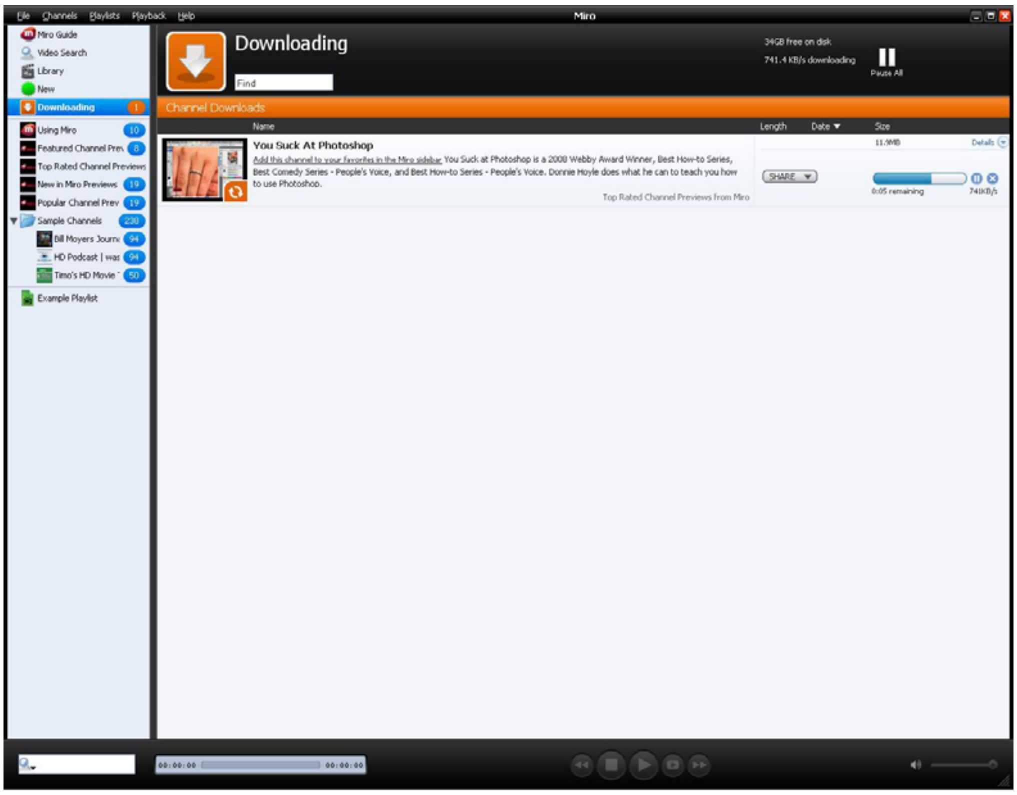
TED Theme: What Makes Us Happy? 10.0MB
Add this channel to your favorites in the Miro sidebar "The Pursuit of Happiness" was 2004's conference theme, but TED speakers over many years have found themselves driven to understand the nature of happiness. Daniel Gilbert's astonishing talk shows how we humans are poor predictors of what will make us happy. Stefan Sagmeister shares how great design contributed to his happiest moments, suggesting that the secret may be attention to detail. And Malcolm Gladwell, tongue in

Tekzilla (Large Quicktime) 296.1MB
Add this channel to your favorites in the Miro sidebar Embrace digital technology. Join the Tekzilla crew and make your tech work better for you. Or you can go live in the woods with an axe. Every Friday Jessica Corbin and Patrick Norton deliver product reviews, computer help, tech tips on everything from iPods to camcorders, HD to the Internet, plus do it yourself projects.

TED Theme: How the Mind Works 57.9MB
Add this channel to your favorites in the Miro sidebar At a conference about ideas, it's important to step back and consider the engine that creates them: the human mind. How exactly does the brain -- a three-pound snarl of electrochemically frantic nervous tissue -- create inspired inventions, the feeling of hunger, the experience of beauty, or the sense of self -- and how reliable is it? Dan Dennett contemplates the mind as an ecosystem in which a new class of entities -- memes -- can

00:00:00 00:00:00

Figure 17 Default interface of Miro Player



Downloading

34GB free on disk
741.4 KB/s downloading

Channel Downloads

You Suck At Photoshop 11.9MB
Add this channel to your favorites in the Miro sidebar You Suck at Photoshop is a 2008 Webby Award Winner, Best How-to Series, Best Comedy Series - People's Voice, and Best How-to Series - People's Voice. Donnie Hoyle does what he can to teach you how to use Photoshop.
Top Rated Channel Previews from Miro
0:05 remaining 741KB/s

00:00:00 00:00:00

Figure 18 Downloading of Video

9.2.2.1 Categories

This section evaluates MIRO software within the following categories, these categories are defined in the section 8 in general terms.

Portability

Miro is highly portable application it supports Windows, Mac , Linux platform providing the necessary support and documentation. The Participatory Culture Foundation creates the Miro player and they provide help and assistant to developers that they want to contribute in the development of MIRO. Currently there is no support for creating a mobile MIRO client.

Code Accessibility

Miro is a free open-source desktop video application that is designed to make mass media more open and accessible for everyone. The majority of the Miro code is cross platform Python with platform specific frontends for Windows, OS X, and GTK/X11. The core of the GUI is written in cross platform HTML. Miro is available under the GPL and is built on top of excellent open-source project such us RSS, BitTorrent, HTTP, HTML, and CSS providing the ability to create a level playing field. The source codes is well documented and the client is well supported by the community of Participatory Culture Foundation.

Scalability and Fault Tolerance

Miro use the BitTornado technology the 3.x branch of BitTorrent inc, Miro initially was using the BitTorrent technology but after they change the license they follow the BitTornado approach. Miro employs RSS (Really Simple Syndication) technology to keep users up-to-date on their favourite video casts. Miro with IPv6 support is not yet available.

Suitability for modification

Miro is well documented and provide an extensive documentation of the source code the use of python provides a stable client and it can be a starting point for the deployment of a P2P implantation in the VITAL++ IMS network testbed.

The developers of Miro have created a web page for helping all the volunteers that developed Miro.

The source code and the developers webpage can be found:

Subversion repository svn co https://svn.participatoryculture.org/svn/dtv/trunk/tv

The code appears to be clearly laid out with a lot of programming comments helping in the creation of a Miro Based application.

SIP & IMS Suitability

- *AAA*: The Authentication model currently implemented by Miro involves the creation of channels rather than true distributed authentication models such as this used in IMS network.
- *Other security concerns including Message Encryption and tamper-proofing:* Torrent files and pieces are checksummed using the one-way Sha1 hash.
- Billing: This is not a feature of the BitTorrent protocol but it may be added to client in order to retrieve downloadable torrents.
- Manageability: The use of RSS provides the ability to manage the content provided by the Miro application.

Scenario Specific Features

- *Use of DHT*: DHT stands for Distributed Hash Table. In Miro the distributed database is based on a UDP based DHT. In fact Miro uses the kademlia implementation supporting the folowing 4 basic operation.
  - PING - used to verify that a node is still alive.
  - STORE - Stores a (key, value) pair in one node.
  - FIND_NODE - The recipient of the request will return the k nodes in his own buckets that are the closest ones to the requested key.
  - FIND_VALUE - as FIND_NODE, but if the recipient of the request has the requested key in its store, it will return the corresponding value.
- Support for isochronous media: Although Miro advertise that it is Internet tv the video must be download before they are been watched.
- *Digital Rights Management (DRM) integration*: Miro main success is that the video that are being shared among the users are created by the authors providing the ability to the users to rate the content and create comments. Producers can upload their original works but they are not able to charge fees.
- *Configurable content replication mechanisms:* From Miro webpage:"*Miro is specifically designed to give video creators and viewers more freedom. We've built Miro to work with as many video hosting sites and video search engines as possible."* Miro creates a RSS feed that consist of any media that users provide providing access to many blogs and video sharing services.

9.2.2.2 Concluding remarks

A major drawback of Miro is that it requires the video to be completely downloaded, before starting playing it. Miro can be a starting point for conveying P2P technology with IMS for the creation of a VITAL++ Client.

9.3 P2P Clients for live streaming

In this chapter we try to focus real P2P client implementations and products and not to current research algorithms or systems that are in progress in P2P live steaming. As we have observed through our market research most widespread and functional clients for live video streaming are proprietary. Due to this reason their analytical evaluation and their use in VITAL++ is infeasible. Although it is important to briefly describe these clients analyze their status, their architecture and their performance in order to be in position to obtain the critical decisions for the architecture and progress of VITAL++. After the presentation of P2P clients for live video streaming we describe some non proprietary clients that deliver other live streaming services (radio, web conferencing etc.) through a P2P architecture and we evaluate them.

9.3.1 CoolStreaming

CoolStreaming is a P2PTV (peer-to-peer television) technology that enables users to share television content with each other over the Internet. The technology behind CoolStreaming is similar to that of BitTorrent. The viewers upload content at the same time the programs are downloaded and viewed. CoolStreaming creates a local stream on localhost and that stream is then read by Windows Media Player, Real Player or other media players. The original coolstreaming code is developed with Python 2.3 on Windows. CoolStreaming is the base technology for Roxbeam Corp., which launched live IPTV programs jointly with Yahoo Japan in October 2006.

Coolstreaming is a pioneering work towards the design of P2P live video streaming systems. Designers describe[30] the way that users exchange blocks and they define an architecture for these systems, and present[31,32] the algorithms for the two major components of such a system. These are:

1. The overlay where is defined the graph that connects the participating nodes and determines the set of nodes that each client uses to exchange video blocks.
2. The scheduler that determines the algorithm that selects the appropriate node and the appropriate block for transmission.

[30] Xinyan Zhang; Jiangchuan Liu; Bo Li; Yum, Y.-S.P.;CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming, INFOCOM 2005.

[31] Bo Li; Susu Xie; Keung, G.Y.; Jiangchuan Liu; Stoica, I.; Hui Zhang; Xinyan Zhang;An Empirical Study of the Coolstreaming+ System Selected Areas in Communications, IEEE Journal on

[32] Coolstreaming: Design, Theory, and Practice Susu Xie; Bo Li; Keung, G.Y.; Xinyan Zhang;Multimedia, IEEE Transactions on

9.3.1.1 Concluding remarks

Through their experience they observe that (1) the churn is the most critical factor that affects the overall performance of the system, and (2) there is a highly skew resource distribution in P2P streaming systems, which has significant impact on resource allocation. They present solutions to deal with these challenges. Additionally they highlight that excessive start-up time and high failure rates during flash crowd, are two of the main challenges any streaming system needs to address.

9.3.2 TVUPlayer

This software is a product of TVU Networks Corporation an Internet company that operates an Internet television broadcasting network that uses P2PTV technology to offer its broadcasters global reach and low costs. Founded in 2005, the company is based in Mountain View, California.

The source code of the player as a product of a company is not available and so we are unable to use it in VITAL++. On the other hand company announces that with the use of its system the broadcaster is able to deliver a lot of useful services:

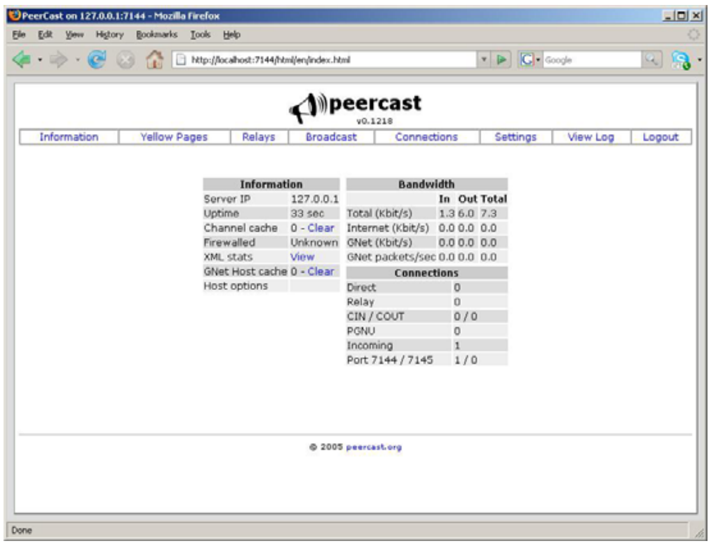
- The bandwidth required to broadcast doesn't increase with the number of viewers, this technology allows TVU broadcasters to achieve massively lower broadcast costs
- It offers broadcasters opportunities denied to them by the limitations of cable and satellite infrastructure. Local broadcasters become global broadcasters; new channels can find a broadcast slot; and bigger broadcasters can create new channels to showcase content that they own but don't have space to broadcast on their existing channels.
- TVU networks' monetization tools allow broadcasters to create subscription channels, pay-per-view events, or advertising-supported channels.
- Content rights management tools allow broadcasters to limit their coverage to specific regions.
- company's TVUAds personalized advertising engine enables seamless in-stream advertising that is targeted to viewers based on their demographics and geographies

9.3.2.1 Concluding remarks

In order to conclude we mention that TVU presents the features that its application claims that offers. These features clarify the motivation towards P2P live streaming and additionally highlights the need for a central management and security services that IMS is able to offer.

9.3.3 Peercast

PeerCast.org was established in April 2002 as a non-profit site providing free P2P radio software. The aim of the project was to create an easy to use, simple and reliable software client that enables anyone to broadcast streaming media on the Internet without the need for expensive servers or bandwidth. PeerCast can serve streams directly to any media player. That means that it can be used in place of a Shoutcast/Icecast server to provide both direct and P2P streaming at the same time. The use of C++ for the creation of application makes Peercast an Operating system independent application. The user interface (see Figure 19) of the software is based on webpage creating an easy to use user interface.

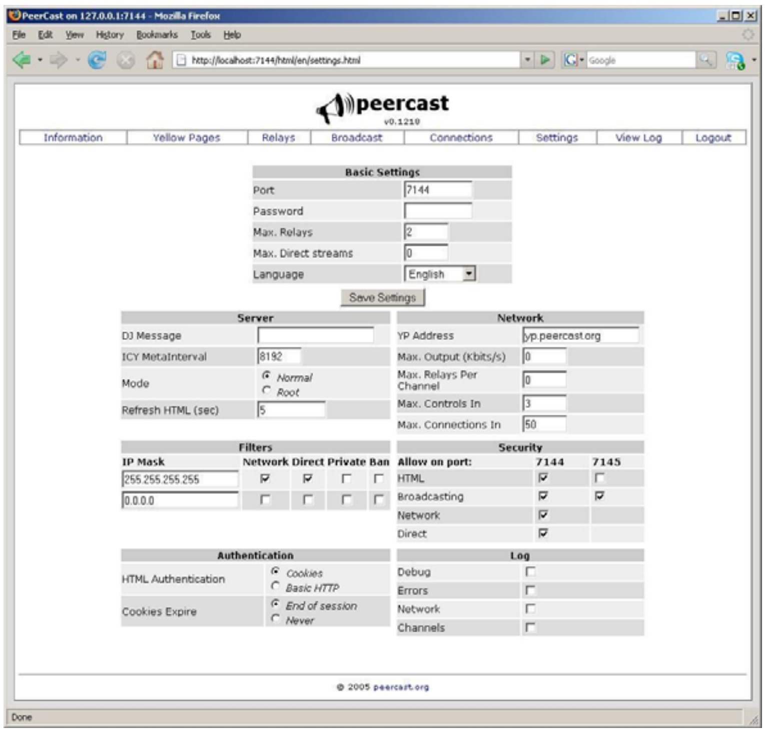

PeerCast on 127.0.0.1:7144 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:7144/html/en/index.html
Google
peercast
v0.1218
Information
Yellow Pages
Relays
Broadcast
Connections
Settings
View Log
Logout
© 2005 peercast.org
Done

Information	
Server IP	127.0.0.1
Uptime	33 sec
Channel cache	0 - Clear
Firewalled	Unknown
XML stats	View
GNet Host cache	0 - Clear
Host options	

Bandwidth	In	Out	Total
Total (Kbit/s)	1.3	6.0	7.3
Internet (Kbit/s)	0.0	0.0	0.0
GNet (Kbit/s)	0.0	0.0	0.0
GNet packets/sec	0.0	0.0	0.0

Connections	
Direct	0
Relay	0
CIN / COUT	0 / 0
PGNU	0
Incoming	1
Port 7144 / 7145	1 / 0

Figure 19 Peercast Webpage


PeerCast on 127.0.0.1:7144 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:7144/html/en/settings.html
Google
peercast
v0.1210
Information Yellow Pages Relays Broadcast Connections Settings View Log Logout
Save Settings
© 2005 peercast.org
Done

Basic Settings	
Port	7144
Password	
Max. Relays	2
Max. Direct streams	0
Language	English

Server	
DJ Message	
ICY MetaInterval	8192
Mode	[✓ Normal] Root
Refresh HTML (sec)	5

Network	
YP Address	yp.peercast.org
Max. Output (Kbits/s)	0
Max. Relays Per Channel	0
Max. Controls In	3
Max. Connections In	50

Filters				
IP Mask	Network	Direct	Private	Ban
255.255.255.255	✓	✓		
0.0.0.0				

Security		
Allow on port:	7144	7145
HTML	✓	
Broadcasting	✓	✓
Network	✓	
Direct	✓	

Authentication	
HTML Authentication	[✓ Cookies] Basic HTTP
Cookies Expire	[✓ End of session] Never

Log	
Debug	
Errors	
Network	
Channels	

Figure 20 Peercast Preference

9.3.3.1 Categories

This section evaluates PeerCast software within the criteria defined above in section 8.

Portability

The PeerCast use P2P technology in order to provide users of the internet to become a broadcaster in that way the traditional costly streaming server are not used and every users shares the radio station that he creates. PeerCast support MS-windows, MAC and LINUX operating systems, there is no mobile client but the use c++ can be the core for a GUI that will be supported from mobile systems.

Code Accessibility

PeerCast is a free peer-to-peer file sharing (P2P) client written in C++. It is under the GNU General Public License(GPL). PeerCast community supports the distribution of modifies clients under the GPL but for commercial use it asks for buing a commercial license.

Scalability and Fault Tolerance

PeerCast is a P2P client that utilizes pure P2P in PeerCast network each user can be a client, server or broadcaster of streams. More extensive PeerCast use the single-tree approach for the creation of multicast distribution creating applicative-layer overlay networks. An example of how PeerCast is like a tree were node are organized as leaves, branches etc. So every branch provide to the leaves the service directly. The node joining and departure strategies used in PeerCast is simple. Node joining request services from the root node A. If A has resources, it will provide service for the node requesting the service and he will provide directly; otherwise, it will redirect.

The same process will be done until the leave will take the service from a branch. Since each node/leave maintains the information of its branch and the leaves that it have an unbalanced tree is constructed. PeerCast has the drawback that when relay is lost the peers that are below of that branch might lose their connection to the stream and must reconnect to another relay potentially causing a skip or repeat in the stream. The mechanisms that are used for the resistance in fault tolerense are random selection, round-robin selection, smart selection according to physical placement, and smart selection according to bandwidth. PeerCast does not support IPv6

Suitability for modification

PeerCast is by a single developer and the latest release is dated 17 December 2007. The source code is available in the following area for download:

Subversion: svn://peercast.org/peercast/trunk

The code appears to be clearly laid out with a well structure and well documented.

SIP & IMS Suitability

- *AAA* :No AAA support
- *Other security concerns including Message Encryption and tamper-proofing:No encryption*
- Billing: No billing schemes
- Manageability: Peercast support Channel and webpage that contains information about the available channels.

Scenario Specific Features

- *Use of DHT*: PeerCast use Tree overlay network approach
- Support for isochronous media: It support the broadcasting of live audio streams. An extension is needed for supporting video. Sip technology not supported by the PeerCast.

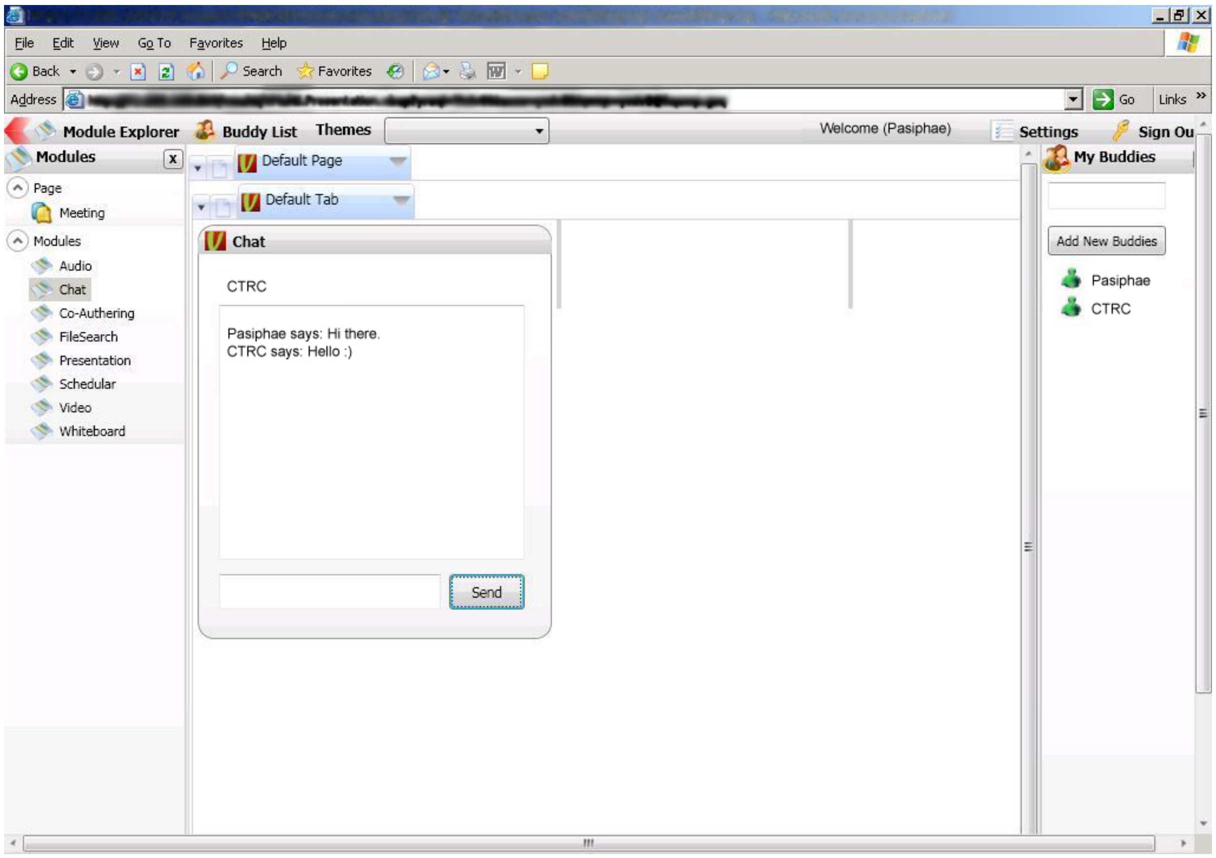
- *Digital Rights Management (DRM) integration*: The channels that are broadcasted are not aware of the Digital Rights Management of the music that they are streaming.
- *Configurable content replication mechanisms:* The use of channel provide in a way content replication mechanisms.

9.3.3.2 Concluding remarks

In PeerCast network each user can be a client, server or broadcaster of streams, by creating application-layer overlay networks. The exploitation of C++ can be the core for a GUI that will be supported from mobile clients. However, the use of the single-tree approach makes it unsuitable for VITAL++.

9.3.4 VMukti - Free VoIP Web Conferencing

Vmukti is a multi-point unified communications, collaboration and conferencing server platform. Its core features include audio, video, chat, file search, whiteboard, file-sharing, presentation, remote monitoring, controlling, sharing, CRM and reports. In the figure below (see Figure 21) the Vmukti client is presented.



Pasiphae says: Hi there.
CTRC says: Hello :)

Figure 21 Vmukti Chat service

9.3.4.1 Categories

The evaluation of VMukti Free VoIP Web Conferencing application is presented in this section. The criteria for the evaluation of the application were described in section 8.

Portability

Vmukti is an open source project that interconnects WEB2.0, P2P technology and telecom operators in an application that provides data video and audio exchange in one portable application. Vmukti supports only Windows Operating Systems. Vmukti support mobile devices[33].

Code Accessibility

Vmukti is in the top 25 ranking on Sourceforge it is under the GNU General public License (GPL). Vmukti is written in C#.net and is available online from the sourceforge currently there is a stable, an unstable and full source code version available for downloading.

Scalability and Fault Tolerance

VMukti use NetPeerTcpBinding Class created from Microsoft inside the .NET 3 frameworks. The use of centralized servers create a secure and reliable network that is resistance and scalable. Vmukti has full supports for IPv6.

Suitability for modification

Vmukti is the first application that converge Web 2.0, P2P technology and telecom world in a unified client that provide a good and reliable client that can constitute the starting point for creating a converged client for an IMS P2P network. The source code is available online:

- http://vmukti.svn.sourceforge.net/viewvc/vmukti/

The code appears to be clearly laid out but it lacks of programming comments on the other hand there is a lot of documentation in:

- http://www.codeplex.com/vmukti

SIP & IMS Suitability

- *AAA*: The Authentication model currently implemented by Vmukti depends in the creation of Bootstrap Nodes. This bootstrap nodes act as a login server for the users providing the starting point for the Vmukti P2P network. This centralized approach can act as a bridge towards the creation of an IMS P2P environment.
- *Other security concerns including Message Encryption and tamper-proofing:* The use of HTTPS protocol provides a reliable interface for the Vmukti client.
- Billing: Vmukti does not support this feature but the use of centralized server with SQL features can help in the deployment of billing service.
- Manageability: Vmukti use centralized Bootstrap nodes providing manageability to the users.

[33] http://www.vmukti.com/main-content/university.html

Scenario Specific Features

- *Use of DHT*: Vmukti use NetPeerTcpBinding from the .NET framework.
- Support for isochronous media: The ability of Vmukti to combine data, audio and video in real time create a powerful client supporting isochronous media.
- *Digital Rights Management (DRM) integration*: Vmukti is a service for video/audio calling so DRM integration is not applicable
- *Configurable content replication mechanisms:* VMukti consists of 3 main entities:
  - Bootstrap Node
  - Supernode
  - Node

Bootstrap Node

The Bootstrap Nodes in the Vmukti P2P architecture acts as a central communication point for the clients of the Vmukti network. Bootstrap nodes depend from the hardware of the computer, the node act as a login server, a Database server, a Web Server and Soft PBX.

Whenever new computer connects to the Bootstrap, Bootstrap decides the type of entity for that node (computer). Bootstrap also acts as a Supernode and a Node.

Functions for Bootstrap are as follows[34]:

1. Login server information management
2. Web server information management
3. Supernode assignment
4. Works as a Super node (services management mention bellow)
5. Works as a Node

Supernode

Supernode is an entity that works as the network relay and proxy server handling the data flow and connections for other users. Supernodes are established automatically by the Bootstrap based on the current network traffic and the capabilities of the user’s machine. Users don’t h the control whether their machines become Supernodes.

A Supernode may optionally alter the client’s request or the Supernode’s response and may serve the request without contacting the specified Supernode. In this case, it would cache the first request to the remote

[34] WhitePaper - Entities of VMukti

server, so it could save the information for later and make everything as fast as possible. A Supernode can serve the requests of the clients without even contacting the specified Supernode, i.e. by retrieving content from the previous saved request made by the same client or even other clients. This is called caching. Supernode locally keeps the copies of frequently requested resources, allowing the VMukti architecture to significantly reduce the upstream bandwidth usage and cost, and increase the performance.

Functions for Supernode are as follows[35]:

1. Node listing and management
2. IPv6, P2P or Http service hosting
3. Message passing between two nodes
4. Dummy client management for Http nodes
5. PBX service management
6. Works as a Node

Node

Every client machine (computer) in the Vmukti network is a node, the node computers typically sends/receives the data from/to requested/required Node. It is also known as communication end-point. The type of the Node is decided based on its characteristics.

- Node with IPv6
- Node with P2P
- Node with Http

9.3.4.2 Concluding remarks

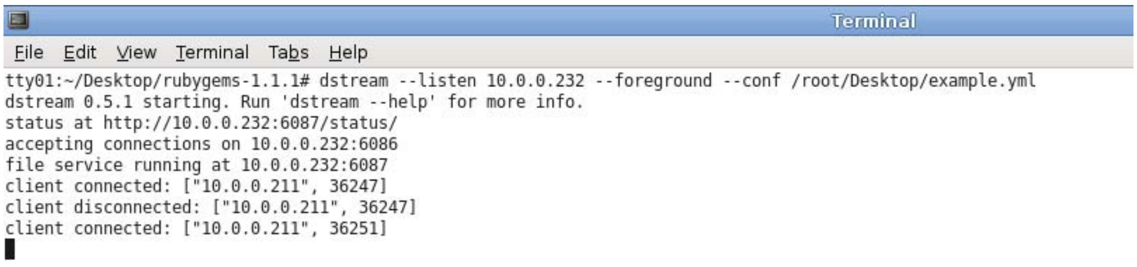
Vmukti is the first application that converge Web 2.0, P2P technology and telecom world in a unified client. The major drawback of Vmukti is that it supports only MS Windows-like systems.

[35] WhitePaper - Entities of VMukti

9.3.5 DistribuStream

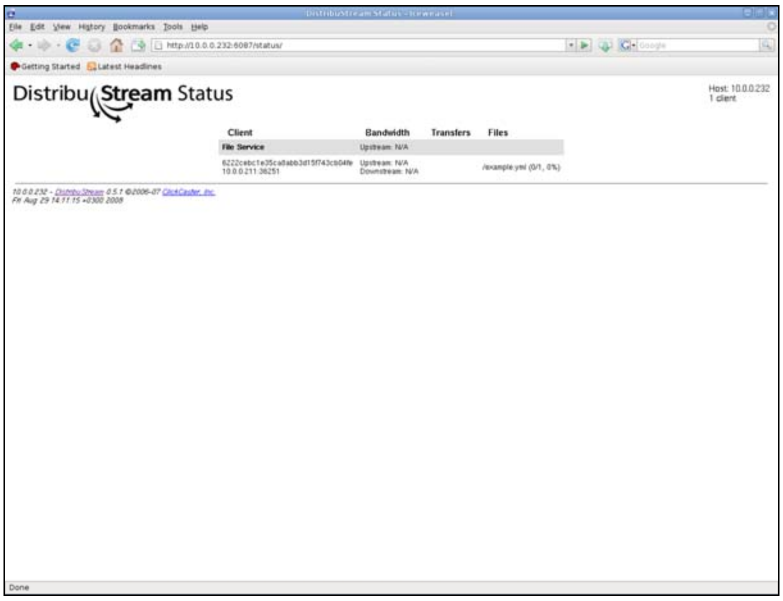
DistribuStream is open source software that implements the Peer Distributed Transfer Protocol (PDTP), a peer-to-peer transfer protocol which utilizes segmented downloading and provides progressive download. Files are received in order and can be consumed for purposes like media playback even as the file is transferring. By using a P2P approach, DistribuStream is able to dramatically reduce the cost of streaming media by partially offloading the required bandwidth onto clients transferring files. This makes it similar to protocols like BitTorrent but unlike BitTorrent, DistribuStream facilitates streaming progressive downloads, making it an open alternative to proprietary protocols like Joost.

The figures below depict a real DistribuStream server running and a client downloading a file from the server.



```
tty01:~/Desktop/rubygems-1.1.1# dstream --listen 10.0.0.232 --foreground --conf /root/Desktop/example.yml
dstream 0.5.1 starting. Run 'dstream --help' for more info.
status at http://10.0.0.232:6087/status/
accepting connections on 10.0.0.232:6086
file service running at 10.0.0.232:6087
client connected: ["10.0.0.211", 36247]
client disconnected: ["10.0.0.211", 36247]
client connected: ["10.0.0.211", 36251]
```

Figure 22 DistribuStream Server


DistribuStream Status - Iceweasel
File Edit View History Bookmarks Tools Help
http://10.0.0.232:6087/status/
Getting Started
Latest Headlines
Done

DistribuStream Status

Host: 10.0.0.232
1 client

Client	Bandwidth	Transfers	Files
File Service	Upstream: N/A		
6222cebc1e35ca8abb3d15f743cb04fe 10.0.0.211:36251	Upstream: N/A Downstream: N/A		/example.yml (0/1, 0%)

10.0.0.232 - DistribuStream 0.5.1 ©2006-07 ClickCaster, Inc.
Fri Aug 29 14:11:15 +0300 2008

Figure 23 DistribuStream statistics Webpage

9.3.5.1 Categories

This section evaluates DistribuStream software within the criteria defined above in section 8.

Portability

DistribuStream is a cross platform application that supports Ms-Windows Mac Linux platforms. There is no client for mobile devises.

Code Accessibility

Distribustream is under the GNU General public License (GPL).

Scalability and Fault Tolerance

Unlike BitTorrent, DistribuStream facilitates streaming progressive downloads, making it an open alternative to proprietary protocols like Joost. DistribuStream is written on top of the Ruby/EventMachine library which allows thousands of concurrent connections.

Suitability for modification

ClickCaster, Inc. and the University of Colorado Computer Science department developed DistribuStream. All source code is available under the GNU General Public License v3.

http://distribustream.rubyforge.org/svn

The code appears to be clearly laid out with programming notes that support of further developing the application.

SIP & IMS Suitability

- *AAA*: No authentication model but there is web based application monitoring that it can support user profiles
- *Other security concerns including Message Encryption and tamper-proofing:* No encryption
- Billing: This is not a feature of the BitTorrent protocol but it may be added to client in order to retrieve downloadable torrents.
- Manageability: This is not a feature of the BitTorrent protocol.

Scenario Specific Features

- *Use of DHT*: DistibuStream relies on emergent, swarm-like behaviour for traffic scheduling.
- Support for isochronous media: The protocol is philosophically different from BitTorrent relying on traffic scheduling. All client/server and peer-to-peer communications are modelled as simple state machines. The

bonus of traffic scheduling is placed entirely on the server. Client/server communication is accomplished with a lightweight TCP, JSON asynchronous messaging format. Peer-to-peer communication is accomplished with HTTP/1.1 using the Mongrel HTTP server. The entire protocol behaves as an ad hoc HTTP caching proxy network.

- *Digital Rights Management (DRM) integration*: The DRM is not supported by DistibuStream.
- *Configurable content replication mechanisms:* DistribuStream utilizes segmented downloading and provides progressive downloads. Files are received in order and can be consumed for purposes like media playback even as the file is transferring creating a live streaming Peer to Peer server.

9.3.5.2 Concluding remarks

DistribuStream is open source software that implements the Peer Distributed Transfer Protocol (PDTP), which utilizes segmented downloading and provides progressive download. By using a P2P approach, DistribuStream is able to dramatically reduce the cost of streaming media by partially offloading the required bandwidth onto clients transferring files. This makes it similar to protocols like BitTorrent but unlike BitTorrent, DistribuStream facilitates streaming progressive downloads, making it an open alternative to proprietary protocols like Joost. The DistribuStream use c++ and it support all operating systems. Distribustream is not suitable for VITAL++, as long as the modifications required for the collaboration of P2P and IMS technologies, are not supported/allowed.

9.3.6 Split Stream

SplitStream is a multicast system distributed as part of FreePastry which was developed and introduced by Microsoft Research (UK), Rice University, (Houston, TX) and Max Planck Institute for Software Systems (Saarbrücken, DE).

The available publications are from 2003. The current release (2.0_04), however, was published in May 2008.

Unfortunately, through the available publications it did not become clear which of the three institutions was involved in which ways and how much. Furthermore, all available publications were either published by the developers themselves or largely based on these publications. External evaluations or experience reports could not be found.

According to the developers, "*SplitStream provides a generic infrastructure for high-bandwidth content distribution. Any application that uses SplitStream controls how the content it distributes is encoded and divided into stripes. SplitStream constructs the multicast trees for the stripes while adhering to the inbound and outbound bandwidth constraints of the nodes. Applications need to (i) encode the content such that each stripe requires approximately the same bandwidth; (ii) ensure that each stripe contains approximately the same amount of information and there is no hierarchy among stripes; and (iii) provide mechanisms to tolerate the intermittent loss of a subset of the stripes.*"[36]

The core message here seems to be that SplitStream is not actually a client but "provides an infrastructure" for streaming and can be used by other applications for exactly this purpose.

SplitStream is largely based or built upon ***Pastry***, a scalable, self-organizing, structured P2P overlay network (see Routing), and ***Scribe***, a scalable application-level multicast system based on Pastry.

SplitStream is strictly focused on splitting files for streaming. It does not seem to have any other features relevant to VITAL++.

Further disadvantages were not reported. This may be due to the fact that all available publications were provided by authors involved in the development of the system.

9.3.6.1 Categories

Portability

As SplitStream is written in Java it can potentially be ported to various platforms, such as mobile phones or interactive TV, etc.

Code Accessibility

SplitStream is part of the FreePastry **Open Source** distribution, which is **part of Pastry**.[37] However, it seems that Pastry and FreePastry are not yet fully developed, let alone robust or fully secure!

"*The initial release of FreePastry is intended primarily as a tool that allows interested parties to evaluate Pastry, to perform further research and development in P2P substrates, and as a platform for the development of applications. Plans for later releases are to provide a robust, fully secure implementation that is suitable for a full-scale deployment in the Internet.*" (Source: http://freepastry.rice.edu/FreePastry/download.html).

[36] Castro et al (IPTPS, Feb 2003)

[37] Available at http://freepastry.rice.edu/FreePastry/download.html

There are documentation files and a tutorial for introduction available at http://freepastry.rice.edu/FreePastry/download.html as part of the FreePastry download package.

Scalability and Fault Tolerance

According to the developers, SplitStream was developed to cater for big and growing peer communities

Forwarding load over all participants using multiple multicast trees reduces the bandwidth demands on individual peers.[38]

The system is able to distribute the forwarding load among the participating nodes, subject to individual node bandwidth limits.[39]

Suitability for modification

SplitStream may be easy to modify, but it does not seem to fulfil more than one of the project's requirements.

SIP & IMS Suitability

SplitStream does not seem to have any relevant features in this respect.

Scenario Specific Features

SplitStream does not seem to have any relevant features in this respect.

9.3.6.2 Concluding Remarks

SplitStream seems to have valuable features for Live Streaming. However, it seems that the Client does not support the other two aspects, i.e. Video on Demand and File Sharing.

9.3.6.3 Related Resources

Abdulla, M.: SplitStream: High-Bandwidth Multicast in Cooperative Environments (October 2004, unfinished presentation document)

Castro, M.; P. Druschel; A-M. Kermarrec; A. Nandi; A. Rowstron and A. Singh: "SplitStream: High-bandwidth content distribution in a cooperative environment", IPTPS'03, (Berkeley, CA/ February 2003).

[38] Castro et al. (SOSP, Oct 2003), p.15

[39] Castro et al. (IPTPS), p.6

Castro, M.; P. Druschel; A-M. Kermarrec; A. Nandi; A. Rowstron and A. Singh: "SplitStream: High-bandwidth multicast in a cooperative environment", (SOSP'03, Lake Bolton, NY/ October 2003).

Kermarrec, A-M. (in collaboration with M. Castro, P. Druschel, A. Rowstron, A Nandi and A.Singh) : Reliable multicast on P2P overlays: Scribe and SplitStream (Cambridge, UK, December 2002)

9.3.7 Tribler (Swarm Player)

Swarm player has been developed based on the Open Source Tribler codebase. Tribler is a highly innovative P2P client, which has been initially developed by Harvard University. The current development team is based in Delft University of Technology and the Vrije Universiteit Amsterdam. This codebase was used as a starting point by the EU-funded P2P-Next Consortium which has added additional features to support live streaming. The codebase is still managed by the original tribler team ensuring consistency and continuity. P2P Next represents a 14 million investment (total project budget is 19 million Euro) in P2P research by the IST programme.[40] Partners include the BBC and the European Broadcasting Union. Unlike other P2P software (Vuze, Miro) which focus on easy access to downloadable contents, swarm player allows a user to simply click on a live .torrent file and tune into any live BitTorrent channel.

9.3.7.1 Categories

Portability

Swarm player is available on Windows, UNIX and Mac OS. Since its beta version has been released for windows and UNIX where as for Mac it shall be released after bug fixing. However there is no version of SwarmPlayer confirmed to be installed on mobile devices. A port to mobile of the core P2P library should be possible as Python is a supported language on Windows Mobile and Symbian Series 60 mobile devices. [41]

Code Accessibility

SwarmPlayer is open source software which uses BitTorrent technology and is building its solution on Tribler[42], an open source windws/Mac/Linux P2P as core

[40] Tribler.org, 19 Million Euro for P2P Research, http://www.tribler.org/P2P-Next/19Million-for-P2P

[41] Nokia launches Python Open Source Programming Language for Series 60-based Mobile Devices, http://press.nokia.com/PR/200501/978226_5.html

[42] Tribler http://www.tribler.com

and the open source VideoLan Client (VLC)[43]. This software is developed using python and released under GNU Lesser General Public Licence. As it integrates two open source technologies already (BitTorrent and VLC) the software design is quite modular. From a brief examination of the code it could be relatively easily modified to change the video playback client.

Scalability and Fault Tolerance

As described in the above section the SwarmPlayer project is in beta however extensive testing has been performed to measure the scalability and performance of this software. The first public trial has been done on on 17th – 26th July 2008. Users were invited from all across the globe to watch a specific live video stream using swarmPlayer. Over the period of 9 days 4555 unique IP-addresses were observed to be tuned in. The scalability and performance measurement has been done on the following parameters.

Performance over time: It has been observed that overall stalling time was fluctuated between 0% and 10%[44]

Pre-Buffering Time: The amount of pre-buffering time determines how long it takes for a peer to start playback. SwarmPlayer has smaller pre-buffering time as it reduces the amount of time the user will have to wait for viewing the first frame. The median pre-buffering time was 3.6 seconds, with 67% of the peers requiring less than 10 seconds, which is 3-10 time shorter than the other deployed systems.

Sharing Ratio: According to definition provided by Mol et al. Sharing Ratio of a peer (or a group of peers) is defined as the number of uploaded bytes divided by the number of downloaded bytes. Peers with a sharing ratio smaller than one are net consumers and those with sharing ratio larger than 1 are net producers. “The trial of swarmPlayer has shown 61% of the IP addresses to be firewalled, and firewalled peers accounted for 52% of the on-line time of all peers”. According to statistic provided by Mol et al during the trial period of swarmPlayer, In total connectable peers have uploaded 0.41 as much they downloaded, while the firewalled peers uploaded only 0.18 times as much. The client’s scalability was assessed by looking at the average sharing ratios against the size of swarm. For small swarms, most peers received their data from seeders and injectors as they are first to obtain each piece and as the swarm grows peers start to forward stream to each other. The average sharing

[43] VideoLan Client (VLC), http://videolan.org/.

[44] “The Design and Deployment of a BitTorrent Live Video Streaming Solution” by Mol et al

ratio improves for the larger swarms. In this respect the SwarmPlayer approach lends itself to large scale broadcast scenarios.

Suitability for modification

SwarmPlayer can be seen as good candidate as a starting point for the P2P-IMS platform suggested by VITAL++ as it is open source with a modular code structure. It also uniquely addresses the issue of live streaming, which is one of the main requirements of VITAL++. Swarm Player evolves the well-supported tribler core.

SIP & IMS Suitability

- *AAA*: Tribler uses the per-user data per-user directory, e.g. on windows this is C:\Documents and Settings\user\Application Data\.Tribler where user is user name. it is authenticated with the private/public key mechanism. Ec.pem file is the private key of the Tribler used, needed to identify himself and ecpub.pem public key of the Tribler used in the PEM format. This is public key is the user's PermID
- *Other security concerns including Message Encryption and tamper-proofing:* In the other P2P software like Vuze security of contents and temper proofing is handled by the hashing techniques. As swarm player handles live streaming a hash cannot be calculated. This may make him vulnerable to the injection attack and data corruption. Data is therefore protected using by using Asymmetric cryptography to sign the data. The injector publishes its public key by putting it in the torrent file. Each piece number is assigned a absolute sequence number and time stamp by its injector.
- *Billing:* This is not a feature of the BitTorrent protocol but it may be added to client in order to retrieve downloadable torrents. Tribler does maintain information about node downloads in decentralised fashion. A collaborative filtering algorithm is used to make recommendations based on previous download history.
- *Manageability:* This is not a feature of the BitTorrent protocol.

Scenario Specific Features

- *Use of DHT*: DHT stands for Distributed Hash Table. In this technique hashes of file are generated which assumes all the data to be known before hand. In case of swarm player it may not be possible for most scenarios, especially when broadcasting a TV channel. Unlike VoD case, peers in live streaming are always playing approximately the same part of the stream. All peers thus require the same data and can not download faster than the stream generated. In existing P2P systems

90% of peers are seeding. In live streaming no peer is ever done downloading, so no seeds exist. SwarmPlayer has developed algorithm to deal with this issue.

- *Support for isochronous media:* SwarmPlayer supports isochronous media by introducing *injectors*. An injector is a unique peer that provides the swarm with the latest generated video and is therefore always online. The video is obtained from the live source e.g. a DV camera. Injector generates a *tstream* file which is similar to the .torrent file but can not be supported by BitTorrent who lack in live stream extension.

- *Digital Rights Management (DRM) integration*: The main emphasize of swarm player is on live streaming. The big broadcasting companies like BBC is working with P2P-Next project which aims to develop open source media delivery platform that is perfectly legal.

Configurable content replication mechanisms:

Swarm Player support 3 kind of behaviours.

- Download the video, and watch it afterwards (typical BitTorrent behaviour)
- Watch the video while downloading it (Video-on-Demand, Vuze and Joost)
- Watch the video while it is being generated (web-cams, live TV broadcasts, etc)

The first two techniques can be found in other P2P clients whereas the third technique is currently unique to Swarm Player. Swarm players use a sliding data window that rotates over a fixed number of pieces. Pieces that fall outside the window will be considered as out-dated. Each peer deletes the out-dated pieces, and will consider them to be deleted by its neighbours as well, thereby avoiding the need of additional messages. When peer joins the network and connects to other peer it learns about which pieces are available and has decide which pieces they own within their own sliding window.

- *Anonymity of peers:* There is no built in mechanism provided by Swarm player to keep peers anonymous however this functionality is not incompatible with the SwarmPlayer/Tribler approach.
- *Rewards mechanism:* The standard bit-torrent choking mechanism is applied.

9.3.7.2 Concluding remarks

Swarm player is open source and its python codebase has a clear and coherent design. It's the result of the extensive research and development of the IST-funded P2P-Next consortium with unique features designed for the demands of live streaming multimedia content. The use of public/private key assymetric cryptography provides strong encryption and mutual authentication mechanisms. The addition of Support for Subscriber Certificates (SSC) is in line with the latest Generic Authentication Architecture (GAA) specifications from ETSI (See 3GPP TR 33.221). Node IP's are not anonymised permitting the geoblocking scenario described. There is no specific support for multimedia conferencing but the live streaming feature of SwarmPlayer could be exploited for group video conferencing. Based on information about the P2P-Next consortium it appears the platform will be extended to support DRM capabilities where required. Tribler has excellent support for custom playlist support and also introduces collaborative filtering to provide recommendations to users about new media they might like. This is suitable for the personalised radio scenario. SwarmPlayer has a very sophisticated media distribution algorithm designed to tackle the problem of seeding during a live broadcast. This mechanism based on a synchronised delivery window for media data would probably be advantageous in remote rural access scenarios as it may be useful in managing variable bandwidth and jitter profiles common outside of urban. areas.
Additionally the software enables video on demand, file distribution or live streaming approaches to content distribution making it highly flexible and an ideal choice for adaptation by the VITAL++ consortium.

10 IMS clients as a basis for VITAL++ functionality

In order to build a hybrid P2P/IMS client, one approach is to expand an existing P2P client, which supports the P2P part and needs IMS extensions. Another approach is to use an existing IMS client and expand it with the required P2P functionalities. In this chapter, we will shortly discuss two IMS clients, namely the VITAL IMS client and the MONSTER IMS client framework. In this context, subsections 10.1 and 10.2 present the VITAL client and elaborates on the enhancements that can be made, while subsection 10.3 presents the Monster IMS client, providing the basis for integrating with the P2P client used and extended in order to support the use cases described earlier in the document.

10.1 VITAL IMS client

The Application Server that has been implemented in the context of VITAL offers three kinds of services:

- Video Streaming
- Data Sharing:
  - Chat Service
  - Collaborative Drawing

Video Streaming

Contrary to standard SIP based applications (video and audio calls), Video Streaming is not bidirectional with respect to the transmission of media content. The SIP part of the communication aiming at call establishment is based on a standard SIP invitation and media negotiation (SDP) session that in the sequel leads to a one way transmission of media packets from the Server to the Client side of the communication. Once the client has successfully registered with the appropriate SIP proxy, the user can invite (SIP call) the Application Server. The invitation as well as the “200 OK” response contain an SDP part in which the two parties define the codec parameters as well as the network ones that should be considered during RTP transmission. The movie selection feature that is offered to the AS clients is based on the exchange of SIP Instant Messages (IM). Extra information is added to these messages, which is aimed at being processed by the UAC and UAS so that the various features of the service can be supported. After the successful call setup the Server sends via IM a list with the available movie clips and a keyword to be used by the client to request the transmission of the selected movie clip. The movie selection request is also sent by the client via IM. The implementation of the Application Server and the corresponding clients was done by using standard SIP techniques so that the solution can be easily integrated with the

IMS platform and be considered as a proof of concept with respect to the additional services (that are usually feasible to be provided over public IP infrastructures) that can be realised over the IMS platform.

Data Sharing

In the Data Sharing applications (Chat and Collaborative Drawing) a registered client can send to the Application Server either a text message or a drawing description via IM. The Data Sharing server builds a list with all the client instances that have communicated at least one Instant Message to it. This list serves as reflector for all the messages that arrive to the server. Every message is distributed to all URIs included in the list with an indication revealing the originator. In case a message sent to a specific URI is not acknowledged by a “SIP 200 OK” the server increases a counter by one until the counter reaches the value of ten. In such case the entry is removed from the list. The originator of a message receives as well the message it has sent which is not displayed on the user’s application GUI unless it has been received from the AS. The UACs translate the received messages according to their type (chat or drawing) and display these on the appropriate GUI element (text area or drawing canvas).

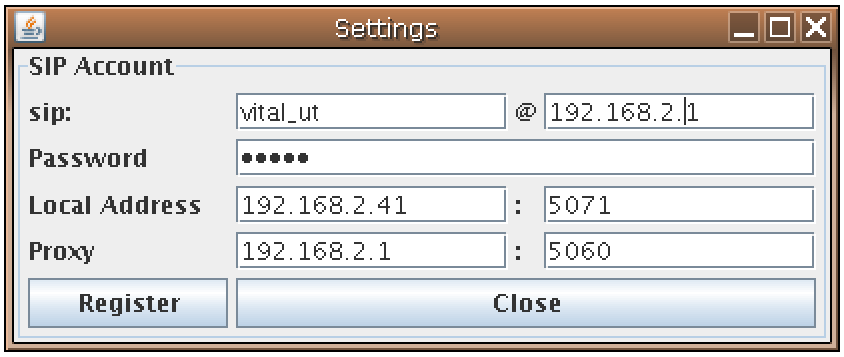
10.1.1 VITAL IMS Client description

Overview

The Client SW is a Java SIP UAC. It contains a SIP handling module and integrates media handling resources from SUN’s JMF package (Java Media Framework). It supports SIP IM (Instant Messaging) and it utilizes it to provide Chat and Collaborative Drawing Services.

GUI Description

In the settings dialog users define their account and the accompanying password. They also define the proxy with which they can register (both address and port number). Users have also to define the address and the port of the network interface through which their host connects to the network. The port number can be anything, in case of definition of an occupied port the client increases the number of the port until it finds a free one. The user presses the “Register” button to register with the proxy, the button then becomes “Deregister” for deregistering.



SIP Account

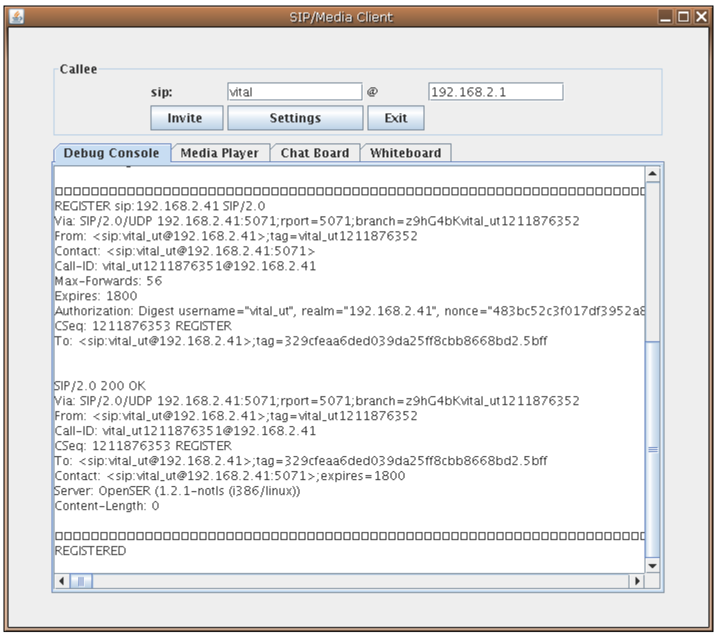
sip: vital_ut @ 192.168.2.1

Password ●●●●●

Local Address 192.168.2.41 : 5071

Proxy 192.168.2.1 : 5060

Figure 24 Settings dialog



Callee

sip: vital @ 192.168.2.1

```
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
REGISTER sip:192.168.2.41 SIP/2.0
Via: SIP/2.0/UDP 192.168.2.41:5071;rport=5071;branch=z9hG4bKvital_ut1211876352
From: <sip:vital_ut@192.168.2.41>;tag=vital_ut1211876352
Contact: <sip:vital_ut@192.168.2.41:5071>
Call-ID: vital_ut1211876351@192.168.2.41
Max-Forwards: 56
Expires: 1800
Authorization: Digest username="vital_ut", realm="192.168.2.41", nonce="483bc52c3f017df3952a8
CSeq: 1211876353 REGISTER
To: <sip:vital_ut@192.168.2.41>;tag=329cfeaa6ded039da25ff8cbb8668bd2.5bff


SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.2.41:5071;rport=5071;branch=z9hG4bKvital_ut1211876352
From: <sip:vital_ut@192.168.2.41>;tag=vital_ut1211876352
Call-ID: vital_ut1211876351@192.168.2.41
CSeq: 1211876353 REGISTER
To: <sip:vital_ut@192.168.2.41>;tag=329cfeaa6ded039da25ff8cbb8668bd2.5bff
Contact: <sip:vital_ut@192.168.2.41:5071>;expires=1800
Server: OpenSER (1.2.1-notls (i386/linux))
Content-Length: 0

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
REGISTERED
```

Figure 25 Main Window – Debug Console

All the SIP messages processed (incoming and outgoing) by the client are printed on the Debug Console.

Movie Playback

Once the user has invited the Application Server, on the Media Player tab a list with the available movies appears. Pressing any of the buttons triggers the server to start streaming the corresponding movie clip. The user can change on the fly the selected movie or restart an already ongoing one by pressing the corresponding button.

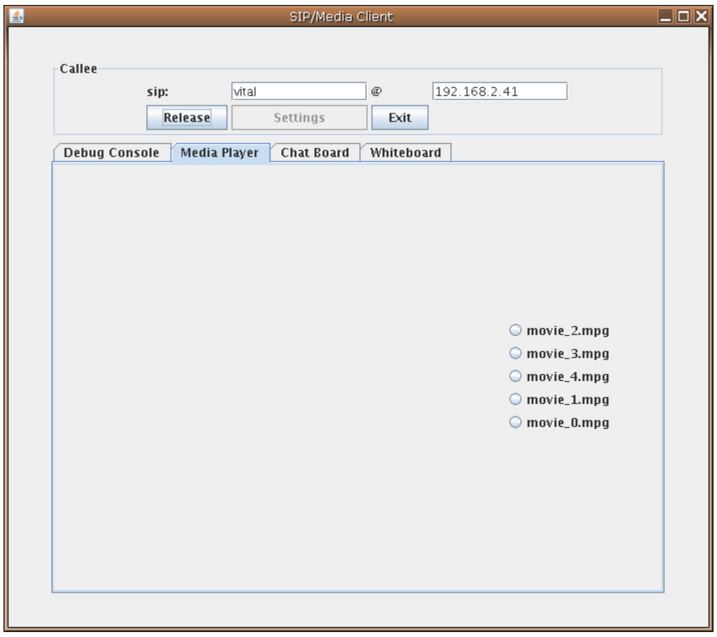


Figure 26 Main Window - Media Player

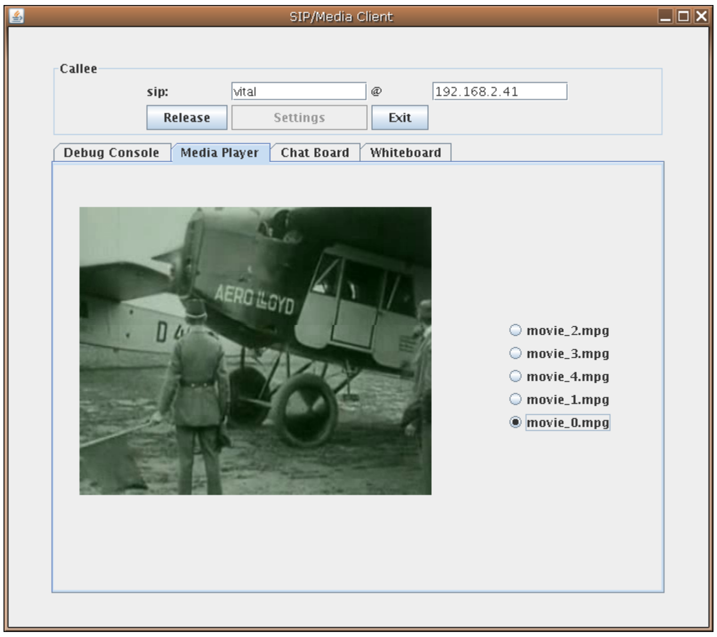


Figure 27 Main Window - Media Player (movie 0)

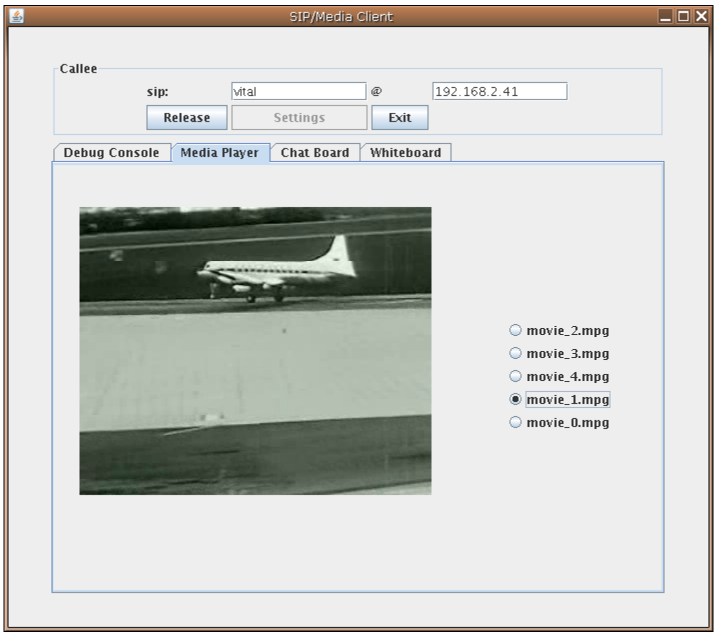
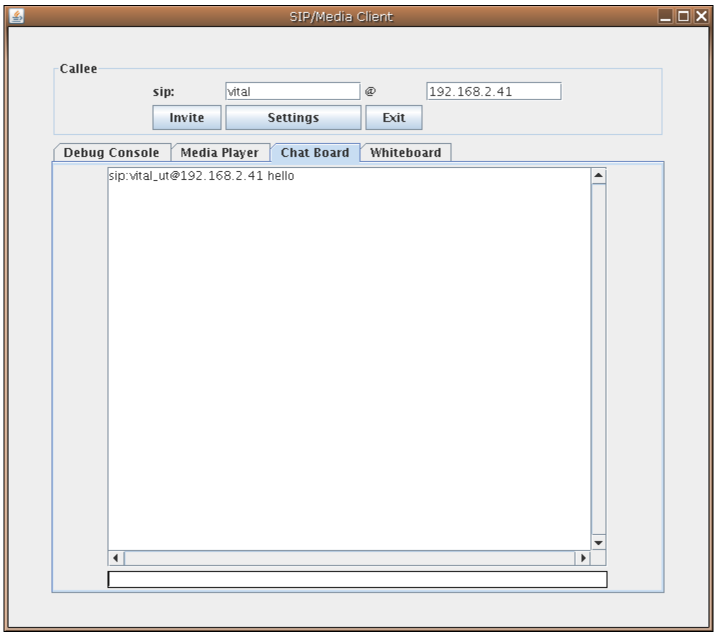


Figure 28 Main Window - Media Player (movie 1)

Chat Service

Using the Chat Board the user can send messages that are reflected to all the users that have already sent a message to the AS. The message submitted (by pressing enter after the message has been composed) is not reflected directly on the chat board of the user but it has to be received by the reflector the AS is implementing.



sip:vital_ut@192.168.2.41 hello

Figure 29 Main Window - Chat Board

Collaborative Drawing Service

Using the Drawing Board the user can draw straight lines (by clicking on the canvas – holding the mouse button and releasing it to another point within the canvas). The drawing instruction is sent to the AS and then it is broadcasted to all the parties that either have sent a text or drawing message. Again the line drawn by the user need to be reflected back by the AS in order to be displayed on the canvas.

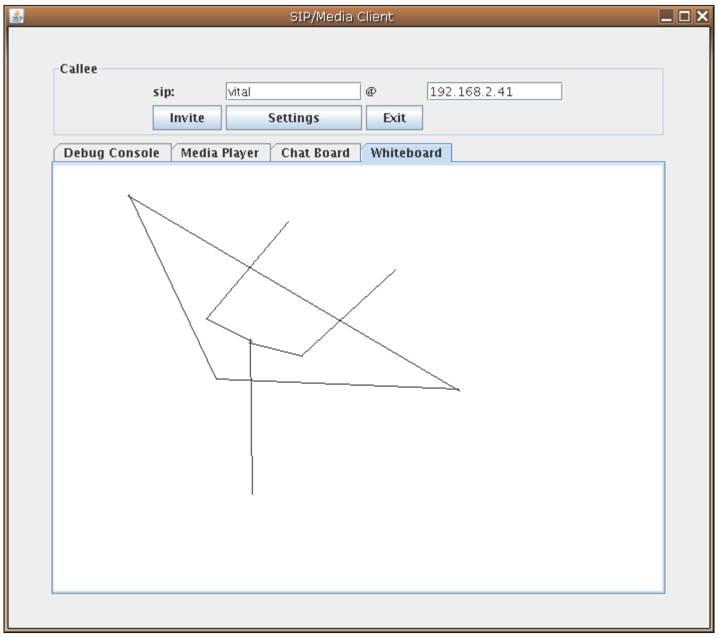


Figure 30 Main Window – Collaborative Drawing

10.1.2 Evaluation of VITAL IMS client concerning compatibility with the VITAL++ use cases

The VITAL client has been developed in the context of the VITAL project and has been tested against compatibility with the IMS operations with traffic experiments on the IMS network of NSN.

To analyze the relevance of this client in serving the operations of the three use cases described in section 7, we need to divide its operations in two parts, the signaling and data part.

Starting from an examination of the three selected use cases, we observe that their basic signaling operations, such as user recognition and data path setup for content delivery do not deviate from the classical server/client communication paradigm and therefore may be realized with standard IMS signaling. Yet, management of content distributed across the network requires utilization of special content discovery and assembly methods suited to the P2P communication paradigm. These methods are not supported by IMS clients.

Furthermore, in IP networks the data part is always implemented with a variety of coding schemes and data encapsulation protocols (e.g. RSVP, RTP, etc) and therefore cannot be formulated into specific categories. However, in regard to content management, all three use cases have the common characteristic of working with fragments of content spread in the network. Gathering and delivering such fragmented content to the user requires implementation of special messages for peer-to-peer communication and real time content assembly algorithms. In the communication model envisaged in section 7.1.2.1 the IMS client does not have to gather the content using such P2P messages and assembly algorithms but retrieves it assembled from a P2P server. This implies that the IMS client should support additional signaling mechanisms to implement communication with the P2P client. Since such mechanisms are proprietary to the IMS operations are not supported by IMS clients.

Having analyzed both the signaling and data parts operations we can make conclusions about the ability of the VITAL client to support them.

As a first conclusion it can be stated that the VITAL client is compatible with the IMS operations required for the integration of the use cases within IMS networks. However, supporting IMS-based operations related to fragmented content discovery and assembly requires further development on the signaling part so that the client can communicate with P2P servers according to the signaling scenarios stated in section 7.1.2.1.

Concerning data part functionality, it can be stated that since the IMS signaling operations are independent of the format of the data traffic, an IMS client may support any coding scheme (data format) without bringing up any impact on the functionality of the IMS network. It is therefore possible for a client to support without modifications any content type, as long the appropriate content encoder and decoder algorithms are supported by the client software.

Following the above analysis the table below makes an extrapolation of the observations made for the VITAL client to all IMS client types to conclude to general statements concerning P2P operations support.

Use Cases/Client supporting functions	IMS signaling	IMS-P2P server communication	Content delivery[45]	Multiplexed content
RBB remote	Yes	No	Yes	-
Soft Radio	Yes	No	Yes	Yes
Remote Rural Areas	Yes	Yes[46]	Yes	-

Table 2: Classification of IMS clients concerning support of the operations of the use case scenarios

[45] 'Yes' is valid only in cases where the client has the appropriate encoder/decoder algorithms embedded.

[46] 'Yes' is valid in some cases, when the use case designates that the content can be taken from sources where it is stored not fragmented.

10.2 The enhanced VITAL client

The client described in this section will be an enhanced version of the client implementation of the VITAL project able to support IMS signalling as well as P2P operations for content management. At its current form, the VITAL IMS client integrates in a single piece of software three user applications:

- A chat client, implementing point-to-point user communication with text messages exchange.
- A data sharing client, implementing sharing of graphical information and real time editing.
- A video streaming client, implementing a video on demand service.

Concerning its signalling part, currently the VITAL IMS client implements a version of the SIP protocol which allows it to communicate with the IMS core networks and perform all typical user operations, such as user registration/deregistration, charging and session initiation/termination.

The data part of the client is based on the adoption of open media delivery frame of the VideoLan open software project (www.videolan.org). The server side required for the implementation of the video streaming service is also based on the VideoLan project.

Both client and server sides of the VITAL IMS client were tested against compatibility with IMS operations on the IMS core network of Siemens, whereas integrated experiments with real content were held with the IMS client installed on the testbed of UoP.

Transformation of the VITAL client into a P2P piece of software will require modifications in the way the client communicates with the IMS network and the data networks containing content. The server should also be considerably modified with functions allowing tracking and assembly of user-requested content residing in the network in form of fragments.

As Figure 31 depicts, the server side will be replaced by a P2P client, which is ‘seen’ by the IMS client as the server of the requested content. The IMS client may communicate with the P2P client with HTTP messages in order to submit a request for specific content delivery. Once the user request is intercepted and recognized by the P2P client, the latter contacts the content tracker in order to get an updated list of peers in the IP network having specific parts of the requested content. Once this list is obtained, the P2P client answers affirmatively to the IMS client concerning its request and the latter contacts the IMS network to get registered and charged for the requested content. Successful user registration triggers the content downloading process on the P2P client. Content assembly is performed either dynamically by the IMS client or locally by the P2P client through utilization of special packet (or fragment) assembly queues.

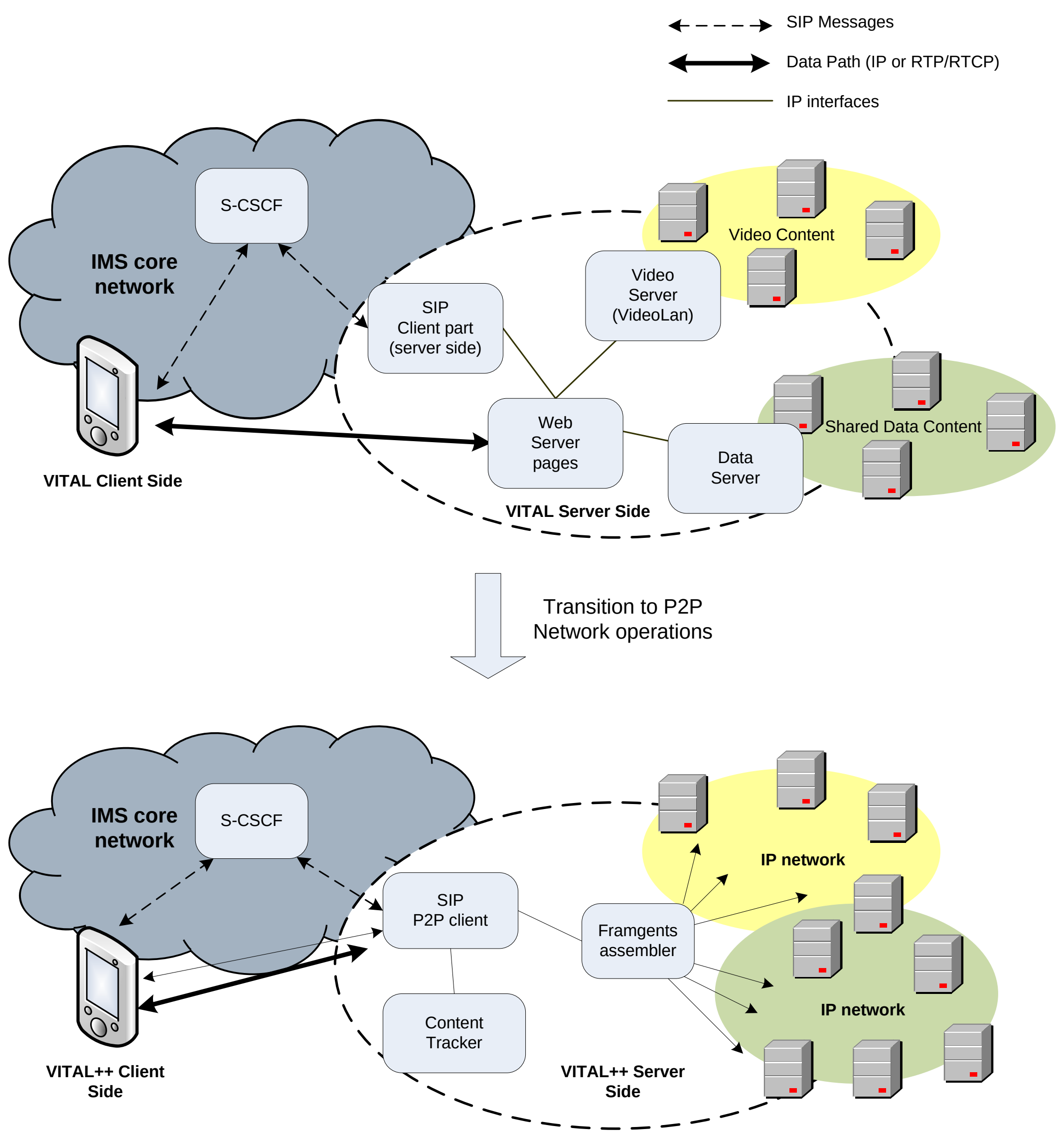


Figure 31 Transformation of the VITAL client/server to match P2P network operations

In its new form the IMS client will support two application types; a) file transfer and b) live media streaming. The first application type will be realized by BCT within the first year of the project and the second during the half project period.

Traffic experiments for the validation and performance evaluation of the two applications will be hosted on the UoP testbed, while service deployment and IMS operations will be provided by the commercial network of Voiceglobe and the IMS testbed of Fraunhofer, respectively.

10.3 Monster IMS client

The MONSTER SDK is a set of independent implemented libraries and components which allows rapid creation of IMS and non IMS client applications.

Figure 32 gives an overview on the general elements of the framework. While the picture is organized in layers, this does not mean the upper layer depends directly on the existence of the lower layer. Each layer provides general interfaces which are intended to be used by either components of the framework itself or third party add-on components. The application framework components will keep track of all layers and combine the components from each one to create an application. So developers will always have the choice whether their application is an instance of the application framework or if it just uses components from one of the layers specified there.

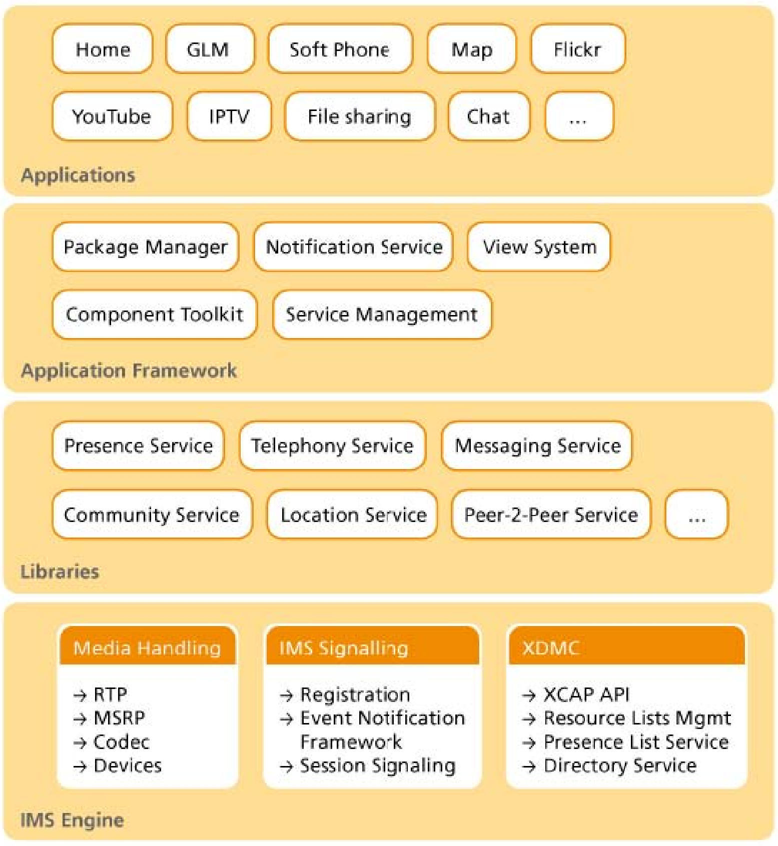


Figure 32 MONSTER framework

10.3.1 IMS Engine

The IMS Engine consists of three main components which are combined to provide fully integrated IMS functionality. Each of these components provides well defined interfaces which allow the implementation to be exchanged easily without affecting the functionality of the remaining components.

Signalling

The signaling component implements three main use cases on top of the SIP protocol: registration, session management and event handling. The registration procedure makes the communication endpoint information from the user available within the IMS core network. All IMS related extensions to the SIP protocol such as the service route header, AKA authentication and network access info header are supported by the signalling component.

The session management part implements creation, modification and termination of multimedia sessions. It controls the media manager for codec and transport negotiation and creation, termination of media streams.

The event management system wraps the SIP event package and provides additional services like automatic subscription handling and event filtering.

Media

The media manager is responsible for creation and management of media connections. Additionally the media manager takes care for protocol and codec implementations, installed within the framework. Currently the following set of transport, protocol and codec implementations are available.

Transports:

UDP

TCP

Protocol:

RTP

MSRP

HTTP

Codec:

Audio: G711, AMR, GSM, MP3, AAC

Video: H263, H264, MPEG2, MPEG4

The current implementation of the media manager relies on the Java Media Framework[47] and can be extended by providing plug-ins for this framework.

Configuration

The configuration component implements access to the shared client configuration within the IMS core network. It contains three parts: a high level API for retrieving and modifying shared xml configuration documents, a notification API for retrieving notifications on document changes and specialized API's for getting access to predefined service configuration as:

[47] Java Media Framework: http://java.sun.com/javase/technologies/desktop/media/jmf/

Resource lists
Directory information
Common Policy Management
Presence authorization

10.3.2 Application Components

The application components layer provides several components to support application development. These components can be used as class library and they offer easy to use, high level APIs.

Presence Service

This service implements the IMS presence event package by reusing the event API offered by the IMS Engine.

Location Service

Location information can be used to implement a lot of useful applications. This service gives access to the actual location of the device the application is running on. The use of location information source allows adaptation to several scenarios – GPS based location, Cell ID based location. If there is no location information available through the network, e.g. for a desktop PC with LAN connection, location information must either be user/administrator supplied or location services will not be available.

Telephony Service

The call service manages creation, modification and termination of voice and video sessions.

Chat Service

The chat service manages creation and termination of text based chat sessions.

Peer-2-Peer Service

Peer-2-Peer Services are currently not implemented in the MONSTER Framework. Due to the modular architecture, it is possible to create additional modules, which then can realize Peer-2-Peer services, i.e. create content related overlays, forward streams, etc. As the MONSTER Framework already features a SIP stack for IMS communication, this can also be used to implement P2PSIP based services with only minimal effort.

10.3.3 Application Framework

The application framework was designed to support the development of feature rich client applications. It offers a flexible structure for new applications and provides a clear application design. Anyway developers are free to choose whether they use this framework or not in order to build their applications and modules

Component Toolkit

The Component Toolkits provides a set of classes for creation of component based applications. It reduces the needs for writing infrastructure code by offering component related services like:

- Component Lifecycle Management
- Class and Service factories
- Component composition through dependency injection

Beside the core services this toolkits offers a clear application design which is well structured. It implements several patterns and combines them to create a framework which can be used to create new applications.

View System

The view system helps to separate presentation logic from application logic. It basically consists of a view class factory which creates views on demand and presents them to the user. The view classes themselves are implemented in a technology specific way and can be adapted to several platforms (Desktop, Mobile, Web) and presentation frameworks (AWT, Swing, SWT, etc.). The view system is well integrated in the component toolkit to support Model View Controller patterns.

Package Manager

The whole framework can be extended through new functionality. The package manager realizes a plug-in concept where the framework can be simply extended by adding new JAR files. The package manager also includes the ability to manage versions and automatic updates of plug-ins.

Device Manager

The device manager offers an OMA DM compliant interface for managing the framework deployment and configuration. It allows to install new framework modules on devices.

Notification / Message Service

The notification and message service enables client application developers to use a presentation independent framework for user notification and user interaction.

The messaging services can be used to display information or error messages with the ability to get feedback while the notification service will notify the user about events (incoming messages, presence changes, ...).

10.3.4 Applications

Based on the functionality of this framework a set of applications were created. These applications range from the well known Softphone functionality till the integration of Web2.0 functionality like Google Maps and Flickr.

Address book and buddy list

The address book allows personal contact list management. Contacts can be organized in different groups. Well known values for contacts like communication address can be stored and modified. The list of values which can be associated with contacts is extensible. The buddy list is stored in the XDMS and is accessible from every device.

The presence service allows the presence state of each contact to be seen and also manages subscription to presence notification of each contact.

Softphone

The softphone is a full featured telephony application on top of IMS functionality. It provides basic functionality as call creation and termination as well as advanced call features as call transfer, hold/resume, video and conference calls2.

Chat

The chat application provides chat functionality which can be found in actual messengers like MSN, ICQ or Yahoo. But in contrast to these applications, this implementation aligns with the standards managed by OMA and 3GPP. Next to the basic chat functionality, also additional services like binary file transfer and chat conferences3 are implemented.

Map

The map application combines the location service with Web2.0 services like Google Maps to offer map functionality. This includes graphical presentation of map material and special locations as the own position provide by the location service. By enriching the presence information with location data, the map application is able to show the buddies on the map.

File sharing

This application enables sending and receiving files to contacts and groups. This is a standard functionality, also known from common messengers, like ICQ, etc.

10.3.5 Evaluation of Monster client concerning compatibility with the VITAL++ use cases

The MONSTER IMS client framework is currently under construction as an in-house project at Fraunhofer FOKUS group NGNI. It consists of multiple layers of APIs for extensions and plugins, residing at different levels of complexity.

To analyze the relevance of this client in serving the operations of the three use cases described in section 7, we use the same scheme as for the VITAL IMS client, which is evaluated in section 10.1.2.

As a first conclusion it can be stated that the MONSTER IMS client framework is compatible with the IMS operations required for the integration of the use cases within IMS networks. However, in order to support conferencing, as described in section 7.1.2.1, an external conferencing server is required. Also, in order to re-assemble different media types for a session (e.g. audio and whiteboard), additional development for the signaling part would be required.

Concerning data part functionality, it can be stated that since the IMS signaling operations are independent of the format of the data traffic, an IMS client may support any coding scheme (data format) without bringing up any impact on the functionality of the IMS network. It is therefore possible for a client to support without modifications any content type, as long the appropriate content encoder and decoder algorithms are supported by the client software.

Following the above analysis the table below makes an extrapolation of the observations made for the MONSTER IMS client framework to all IMS client types to conclude to general statements concerning P2P operations support.

Use Cases/Client supporting functions	IMS signaling	IMS-P2P server communication[48]	Content delivery[49]	Multiplexed content
RBB remote	Yes	No	Yes	-
Soft Radio	Yes	No	Yes	Yes[50]
Remote Rural Areas	Yes	No	Yes	-

Table 3: Classification of IMS clients concerning support of the operations of the use case scenarios

[48] 'No' is valid as currently no P2P protocols or algorithms are currently available for MONSTER.

[49] 'Yes' is valid only in cases where the client has the appropriate encoder/decoder algorithms embedded.

[50] This just means to change the media relation by using IMS session signalling.

11 Overall Evaluation and Design Goals for P2P and IMS clients

In this deliverable we have described and evaluate P2P clients and IMS clients that exist in the market right now. Additionally in the beginning we have presented the applications that VITAL++ will deliver (P2P content distribution (CD), P2P live streaming (LS) and P2P assisted video on demand (VOD)) and the current research challenges that we have to address in order to achieve these goals. Through our evaluation we have spotted the limitations that today's P2P clients have which gives and motivation and highlights the issues in which we have to focus.

At first the major limitation that we have observed in every P2P system that we have analyzed is the **absence of a mechanism that will ensure content availability**. On the other had here are two architectures that address P2P *content location problem* in today's P2P clients. The first is the use of a flooding mechanism that we observe in clients as Limewire and Cabos that use the Gnutella network. Each user that requests content issues a query in the system in order to retrieve the network addresses of other users that have the content. This query is flooded through the overlay and every node that owns it returns to the initiator of the query its network address. Another approach that we have seen in Azureus and Tribler is the insertion of the content as a key in a DHT. Each node that owns content inserts this information in the DHT and each node queries the DHT in order to retrieve it.

What is needed is the extension and enhancement of the existing system for content location to ensure content availability. In more detail there is a need for a monitoring system that will detect the existence and the redundancy (frequency) of each content block in the participating users. This system must be supported by a centralized manageable content storage system that will cooperate with the DHT in the P2P *client* and it will guarantee the data availability and will dynamically infuses to the users rare or absent content blocks.

The second limitation of the P2P clients that we have evaluated is the **absence of uninterrupted service functionality, especially for real time applications as LS and VoD.** This is due to the insufficient and dynamic network resources that are unable to meet the requirements of these applications. Coolstreaming and Pplive try to optimize the graph of the overlay, create a sophisticated scheduler and exploit the heterogeneous bandwidths in order to deliver P2P LS. From their poor QoS it is clarified, from our point of view, the usefulness of a centralized system that with the use of video encoding and/or transcoding will adapt the service requirements to the network or user resources that it will dynamically monitor. Additionally it will have the ability to supply temporary (for a short time interval) or spatially (in specific network regions) network resources in order to stabilize the service in case of aggregate bandwidth fluctuations and high rates of incoming users (flash crowds).

For peer assisted VoD the research problem becomes more challenging as we observe through the examination of clients as Azureus and Tribler and recent research works. The major reasons for this situation have described in the previous paragraph. Additionally except the adaptation in the network resources also rises a research challenge on the creation of more sophisticated overlays, part <u>of these schedulers are already implemented in some P2P clients</u>, in order to have a high level of cooperation of users with common content interests and more advanced schedulers in order to satisfy the more strict constraints that service nature introduces (different users consume different parts of the stream in each time instant).

In every P2P client as long as it is not proprietary and it is not designed for commercial use the insertion of a node is performed from a bootstrap node and nodes join the overlay without any restrictions. So no authentication is performed, as no subscriber base exists, so everybody can join an overlay and use its services. On the contrary in VITAL++ **there is a need for an authentication system where IMS will control the entrance of a node in the overlay and will authorize the connection between any participating nodes**. The creation of a connection in the overlay and the termination of a connection will be reported to the IMS authentication system and so bad user behaviour (ex. free riding) will be infeasible.

Additionally in existing P2P clients the content is not protected and every user has access to acquire and even modify the content that it downloads. Furthermore due to network or other component failures content may be corrupted. At last possible attacks can pollute the content and make it unsuitable for consumption. Due to these reasons there is a **need for a content integrity and management system** that will ensure the reliability of the content, it will manage its digital rights and it may be responsible for the key management of encrypted content, <u>with respect to the P2P client's encryption functionalities</u>, in order to develop a content charging mechanism.

At last in order to minimize the load (and also the ISP cost) the is introduced to the underlying network through P2P applications and increase the fault tolerance and the consistency of the services there is a need for a locality P2P overlay. Most <u>P2P clients are agnostic to locality</u> except some proprietary as PPlive that takes partially locality as criterion of the overlay construction and Azureus that uses a mechanism, called Vivaldi, which through distributed optimization creates a vector that reflects the position of each node in the underlying network.

Looking at the IMS clients currently available it is evident that in order to introduce P2P in IMS or in a telecom operators environment at large, IMS and P2P clients are required to intercommunicate with each other for exchanging relevant information towards a common goal. The kind of information exchanged is dictated by the use cases defined in VITAL++ and includes information relevant to security, digital rights, authentication etc necessary for P2P to function in an IMS environment. This can be achieved by means of either integrating the P2P functionality in the IMS client or by defining the proper interfaces between the two types of clients. The latter is the most efficient way as it allows using a variety of P2P clients for deploying different kinds of overlays customised for certain applications.

Furthermore, exploiting such interfaces the IMS clients may use the content distribution capability offered by the P2P clients in order to support a large number of related applications that are currently offered via conventional ways.

This will be the approach adopted in VITAL++ whereby the two IMS clients (VITAL and Monster) will be adapted to interface with the P2P clients and corresponding functionality designed for VITAL++.

12 Conclusions

In this document we have carried out a study of evaluating P2P and IMS clients in the context of a number of use cases. Our objective has been to identify the key features that are currently missing in order to support content distribution in an IMS environment.

We have examined the requirements from the three applications that VITAL++ will deliver as commercial services. Towards this goal we consider that P2P architecture will be of high importance in order to infuse scalable and stable behaviour to them. Additionally IMS will offer a management system that will guarantee a secure, reliable and uninterrupted behaviour to the aforementioned applications.

Through the examination of the architecture and the algorithms that constitute the state of the art in today's file sharing and content distribution P2P systems we have derived useful observations in order to select the enhance the appropriate P2P clients. As a conclusion we can highlight that today content distribution mechanisms and systems achieve a high level of performance in resource utilization and have high levels of fault tolerance. On the other hand content availability, free riding and flash crowds are the major problems of them. Through the combination with an IMS client and the development of a management system we plan to address these technical challenges.

P2P live streaming is a more complicated problem and its major difficulty is that it requires real time constraints in block delivery. These constraints tighten the requirements in the overlay construction and in the scheduler that we have to develop in order to deliver this application. Again data location and availability is an issue that we have to address, the dynamic adaptation of the service resource requirements, to the existing network resources is also of major importance for P2P live streaming uninterrupted functionality. Most P2P clients for LS are proprietary but in the literature we can find promising algorithms and architectures that convince us that the implementation of such a client is feasible and with no very high levels of risk.

The third application is P2P VoD and as we have spotted from the literature is the most demanding application in terms of the research challenges that we have to address. So practically feasible P2P VoD systems are focus on the support of a server system infrastructure as a fail-over solution. On the other hand they can significantly reduce the bandwidth that servers consume by optimizing the piece selection strategy and by the formation of overlays according to the users correlation in terms of requested blocks.

Through the evaluation of P2P clients we have observed that Azureus, and Tribler are open source clients with high level of stability and widely adopted from today's P2P users. Additionally they offer a DHT, many security functions, encryption and even algorithms that exploit locality in the network between users. Additionally they deliver very efficient schedulers for content distribution and also allow VoD features in case of efficient network resources. This makes them strong candidates for their use in VITAL++ architecture.

At last in section 11 we briefly described the major technical objectives that we have to address and the major systems that we have to develop during the duration of the project.

- End of document -