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	**Overall Assessment of the VITAL++ Communication paradigm**

Instrument:	STREP
Theme:	ICT-2-1.6
Report Due:	M30
Report Delivered:	
Lead Contractor for this deliverable:	UoP
Contributors to this deliverable:	Konstantinos Papanikitas (UoP); Jens Fiedler (Fokus); Konstantinos Koutsopouloos (BCT); David Flórez (TID); Shane Dempsey (WIT); Evangelos Markakis(CTRC), Evangelos Pallis(CTRC), Nikolaos Polyxronakis (CTRC)
Estimated person Months:	23MM
Start date of project:	1st June 2008
Project duration	33 months
Revision:	Version 0.4
Dissemination Level:	PU – Public
Internal Reviewer	Evangelos Markakis (CTRC)

Table of Contents

This page intentionally blank

List of Figures

1 Introduction

This report aims the findings of the test that took place during VITAL++. The tests are reported in detail in D5.2. We divide this deliverable in chapters according to the different functionalities that we developed. Finally in the extended conclusions section we present the advantages and the disadvantages that our architecture has.

2 Assessment of Vital++ Sub-Architectures

During the evaluation of VITAL++ architecture we revealed the advantages and the disadvantages of each component. In this section we present numerical results on the performance of each sub-architecture and we comment on them.

2.1.1 Peer to peer authentication P2PA-SA

This section contains the tests to be performed in order to check out P2PA proper working.

2.1.1.1 Single Entity Performance Test

In this test, we measure the delays for the P2P-Authentication message exchange during the initial credential exchange when a peer attaches to the network. Also, the timing for exchanging signed messages are measured and differentiated about message transfer and signature verification. The single steps for which the processing time is measured are as follows:

- IMS Registration.
- Vital++-Server certificate retrieval.
- Client certificate authorization.
- Client-to-client signed message exchange.

The Setup we are using is depicted in the figure below.

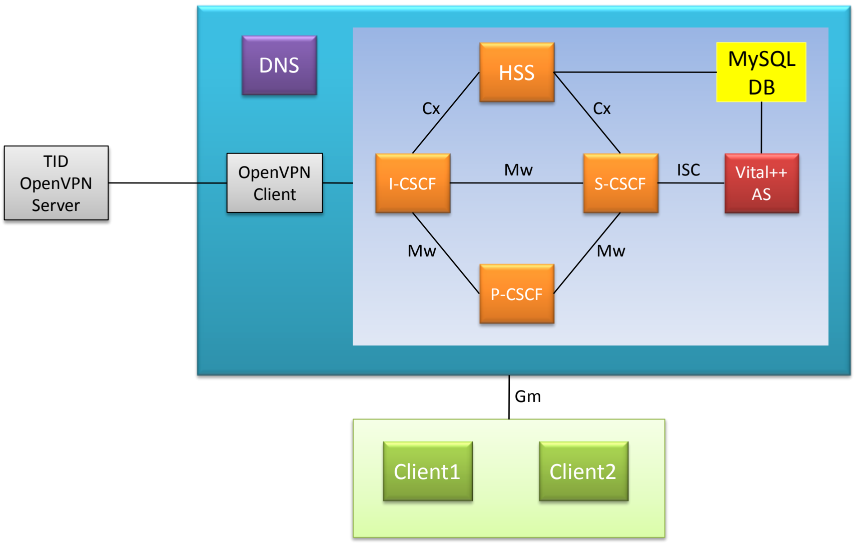

Figure 1: P2P Authentication Testbed Setup

During the test, both clients register with the IMS Core, receive the server certificate and also their own client certificate. Then, client1 transmits a signed message to client2, which verifies the signature. Then, both clients unregister from the IMS Core and terminate. In order to obtain relevant values, the test has been performed 100 times and the delays have been averaged.

The following table presents values for the minimum, maximum and arithmetic average of the measured values in milliseconds.

	IMS Registration	Server Certificate	Client Certificate	Signed Messaging
Maximum	1133.00	41.00	155.00	223.00
Average	207.94	0.48	49.05	40.51
Minimum	70.00	0.00	19.00	14.00

Table 1: P2P Authentication Delays, Single Client

Interpretation of results

For the **IMS Registration**, we see that the average of 207.94 milliseconds is in the expected range, as it involves two registration transactions, one initial and another one with IMS authentication credentials.

The **Client Certificate** exchange is also in an expected range, as it involves only one 'MESSAGE" transaction between the client and the server. If this transaction would be additionally encrypted (which is advised), the value would probably rise by a factor of about 2.5 due to the additional signaling (one more transaction) and processing effort.

Also the **Signed Message** latency looks as expected, as it also involves only one transaction between the clients. It is faster than the Client certificate exchange, as the message is not routed through the IMS Core.

For the **Server Certificate** exchange, the values look by far too small, so this needs further explanation. The sending of the server certificate is triggered by the IMS registration process, i.e. the application server receives a notification about the registration already when the final SIP "200 OK" is being send to the client. Thus, processing in the AS already begins while that message is still on the route to the client, indicating partly parallel processing of both operations. So, the server certificate follows immediately upon the finalization of the registration to the client. This behavior shows that waiting for the server certificate does not have a significant impact on the client, which has to wait a maximum of 41 milliseconds before it can continue operation.

As a conclusion, the extended sign on procedure caused by the certificate exchange is marginal in terms of time at startup and will not interfere with the other tasks a user endpoint will perform at that stage. Also the messaging delay is quite normal compared to the other messaging methods.

2.1.1.2 Server Performance Test

Procedure of Testing

The purpose of this test is to evaluate the performance of the Vital++-AS for P2P-Authentication messages. We measure the delay between sending a message to the server and receiving the corresponding answer. The setup for this test is depicted in the following figure.

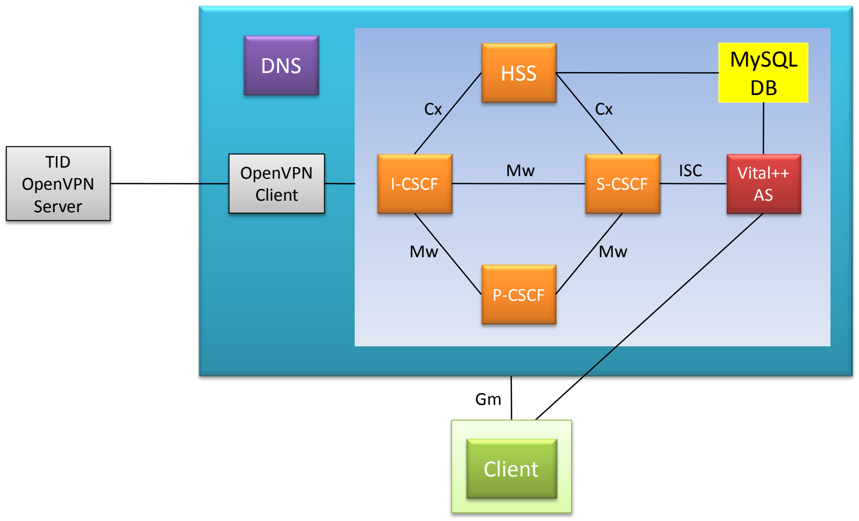

Figure 2: Server Performance Test Setup

The client connects normally through the Gm interface to the core and receives the server certificate through the core. In order to measure the performance of the Vital++-AS and not the performance of the IMS core, the client will send the stress test messages directly to the Vital++-AS. The messages we send are the normal Vital++ P2P authentication messages that are used to sign an empty client certificate.

We always send 1000 request messages at a constant frequency (requests per second), measure the delay until the answer arrives and calculate an average delay for that frequency of requests. The frequency at which we are measuring the performance starts with 10 requests per second and goes up to 1000 requests per second. The result of this test is a graph, which shows how the average delay is related to the frequency of requests being sent to the server. The graph is presented in the following figure.

Interpretation of results

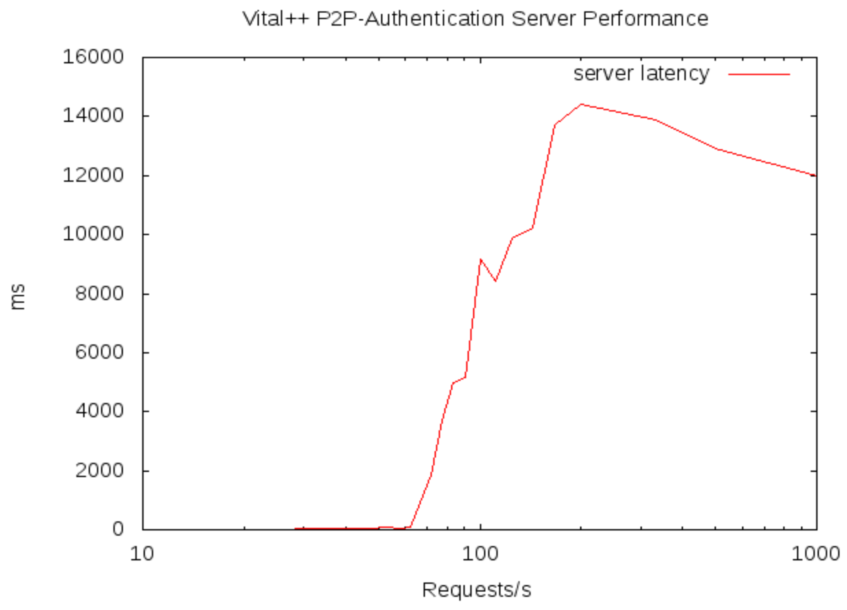

Figure 3: Vital++ Server Average Latency

In this graph, we see that the latency stays in an acceptable range, i.e. in the range of 18 to 50 milliseconds, as long as not more than 60 requests per second are being sent to the server. From that point on, latency literally explodes and the average latency reaches values of up to 14.000 milliseconds. Extreme single values of over 27.000 milliseconds have been observed in that area too.

As a conclusion, the server delay will stay acceptable as long as not more than about 60 users sign on in the same second, which is quite unlikely, but depends of course on the total number of subscribers and the rate by which they register with the IMS core.

2.1.1.3 P2P Authenticated Message Exchange Test

In this test, we want to verify the correct detection of bogus signed messages. We use up to three clients, which are connected to the IMS core and thus to the Vital++-AS. The clients register with the core and the Vital++-AS, i.e. they receive the server certificate and exchange their own client certificate. The private key associated with that client certificate is then used to sign the messages which are exchanged between the clients. Each client sends a fixed number of messages to each of the other clients, corrupting about 50% of the messages on purpose. This is achieved by first signing the message, then changing it before sending it out together with the certificate and signature to the recipient. The recipient tests each message for authenticity and counts the number of bogus and correct messages, while the sender does the same for the number of transmitted messages. Each sender is also a receiver and performs both tasks. The testbed setup is depicted in the next figure.

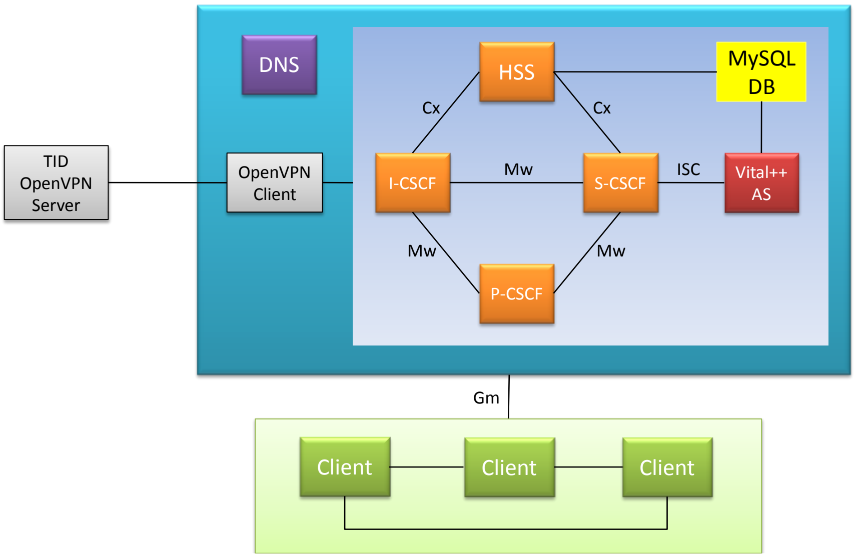

Figure 4: Authenticated Message Exchange Test Setup

Each client sends 500 messages to each other client, from which about 50% are corrupt while the remaining messages are all correct, i.e. authentic. A random number generator determines if a message is going to be corrupted or not. The random number generator has been programmed to produce true/false values with a bias of 0.5, i.e. 50% probability for each value. The test has been performed with only two clients and a second time with three clients. The following table presents the results for the test with two clients.

User	Send correct	Send corrupt	Received correct	Received corrupt
Alice	249	251	252	248
Bob	252	248	249	251

Table 2: P2P Authentic Messaging Results - 2 Clients

From this table, we see that each client has send and received 500 messages and that all corrupt messages have been detected as such, while the correct messages have also been recognized as authentic. This also indicates that no packet loss has occurred.

The test has been repeated with three clients, behaving in the same way, i.e. each of them sends out 500 messages to each other client with a probability of 50% to be corrupt. The next figure depicts the client interaction in the 3-clients case, while the next table presents the result of that test.

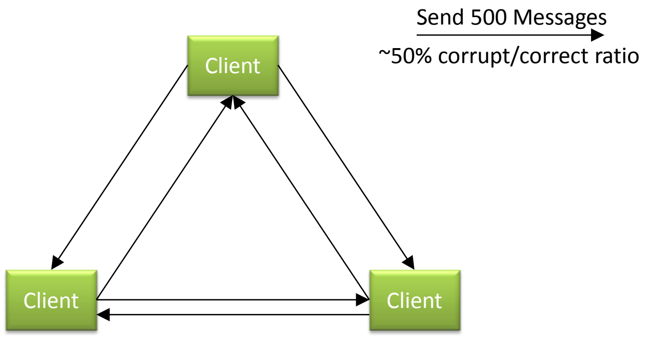

Figure 5: 3-Clients Interaction

User	Send correct	Send corrupt	Received correct	Received corrupt
Alice	468	532	513	487
Bob	502	498	486	514
User.11	499	501	470	530

Table 3: P2P Authentic Messaging Results - 3 Clients

Here we see that each client sends out a total of 1000 messages (500 to each of the other two). A total of 468+499+502=1469 authentic messages have been sent and 513+486+470=1469 have been received and verified as authentic. Also 532+498+501=1531 corrupted messages have been sent out and 487+514+530=1531 messages have been received and been detected as corrupt. Thus, a total of 1531+1469=3000 messages have been exchanged, which equals the intended number of messages to be exchanged (3*2*500=3000 messages from 3 clients to 2 other clients). This also means that no packet loss has occurred.

As a conclusion, the detection of corrupt messages works well, as all corrupt messages have been detected in both test cases.

2.1.2 Content Security CS-SA

2.1.2.1 Result presentation

WIT's Content Security Subsystem Architecture (CS-SA) was tested using 2 separate sets of tests, one exercising accounting functionality and the other without.

The test cases are described in D5.2 as

- Licensing Performance Without Accounting
- Aggregate Licensing and Accounting Performance.

A description of the tests is shown below.

Licensing Performance Without Accounting

A number O(10^3) of "Fair Use" licensing requests are made to the licensing service using the web interface. The total completion time is registered and an average is determined. The Apache JMeter tool is used to generate the traffic and log the results permitting an analysis of the produced data to determine mean and deviation of the request results, demonstrating the performance and scalability of the implementation.

The purpose of this test is to determine the performance of licensing, representing an end-to-end exercise of the functionality of the Content Protection Subsystem, except the Accounting subsystem. A fair-use license is requested as this does not require the generation of user charging information and hence, does not exercise accounting functionality which is dealt with separately in the following test.

Aggregate Licensing & Accounting Performance

A number O(10^3) of content licensing requests are made to the licensing service using the web interface. These requests are of a type and for content that requires a content charge to be calculated for the licensing transaction. Data is recorded for each request that can be used to analyse the system's performance. The consistency of the licensing and charging data is checked.

The Apache JMeter tool is used to generate the traffic and log the results permitting an analysis of the produced data to determine mean and deviation of the request results, demonstrating the performance, scalability and consistency of the integration of accounting and licensing within the associated Subsystem Architecture implementations.

The purpose of this test is to determine the performance of licensing, representing an end-to-end exercise of the functionality of the Content Protection Subsystem including the Accounting subsystem.

Reliability & Performance Metrics

The reliability and performance metrics for both sets of tests are the same as we're effectively testing the same system but with additional functionality exercised in the second test.

Our reliability metrics are:

- The unavailability rate corresponding to the number of requests which are not responded to;

- The error rate where the response is determined to be incorrect, bearing in mind that a valid response is easily checked using a text/string compare function.

Our performance metrics are:

- Mean and standard deviation of latency for the requests;
- Scalability by comparing performance with different numbers of clients making concurrent requests e.g. 1, 5, 10, 50;
- Frequency analysis of latency/request and plotting of cumulative frequency distribution.

System Under Test

The System under test consisted of 2 virtual machines installed on a SUN Fire, running under a VMWare ESX Hypervisor.

	Application Server	**Certificate Authority**
Operating System	Ubuntu Hardy 8.04 - x86	Ubuntu Karmic 9.10 - x86
Kernel	Linux 2.6.24-24-server	Linux 2.6.31-14-generic
CPU	Intel(R) Xeon(R) CPU X5355 @ 2.66GHz	Intel(R) Xeon(R) CPU X5355 @ 2.66GHz
Load Average	0.10, 0.03, 0.01	0.05 0.02 0.01
Disk	9.5GB. Used: 4.3GB, Free: 4.8GB	19GB. Used: 2.9GB, Free: 16GB
Memory	Total 2GB, used: 1534MB, free 492MB, buffers 0	Total 1024MB, used 354MB, free 647MB, buffers 0

Presentation of Results

The JMeter tests were analysed to yield statistical data for the calculation of reliability and performance metrics. The results of both licensing test cases were near identical. This led us to believe that the accounting subsystem, which uses non-blocking input/output where possible, does not add significant latency to the processing of CS-SA requests. Therefore we'll consider the more realistic "aggregate licensing and accounting" test cases for the remainder of this section.

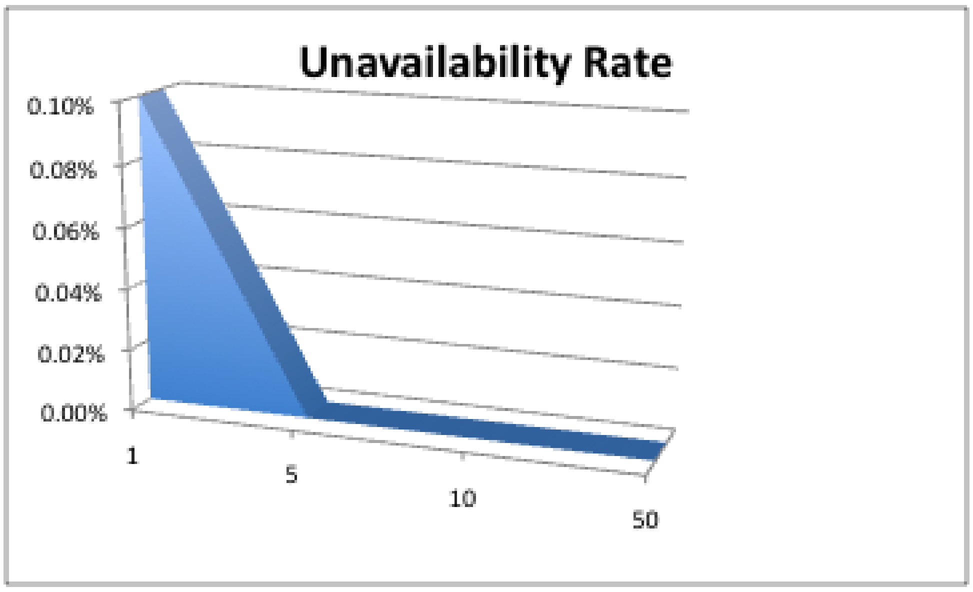

Figure 6: Unavailability Rate

The peak unavailability rate is 0.1% or 1 request failure in one thousand when testing with one client. The availability rates for 5, 10 and 50 clients are 100%.

The error rate for each of 1, 5, 10 and 50 clients was 0%. This means that, of the responses received, none of these responses were incorrect based on comparing the received result with the expected result. Charging data associated with each licensing request was also compared.

A graph of sorted latencies for 1, 5, 10 and 50 clients for a sample of 1000 licensing requests is shown below.

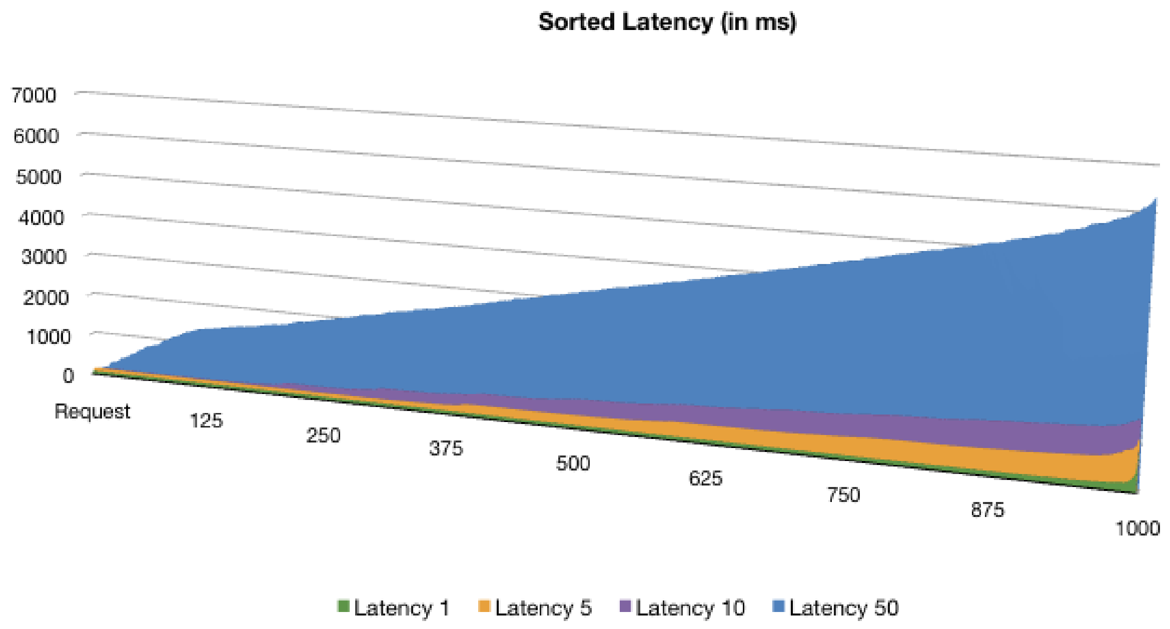

Figure 7: Sorted Latency

The y axis is the latency in milliseconds between request and response. The maximum latency for the system is approximately 6.3 seconds to license and micro-charge a specified content item with 50 clients making concurrent licensing requests. It can also be observed that latency figures rise linearly (approximately) for each set of experiments.

The mean and standard deviation of our latency data is shown in the following graph. Again, the results are calculated for 1, 5, 10 and 50 clients making concurrent requests.

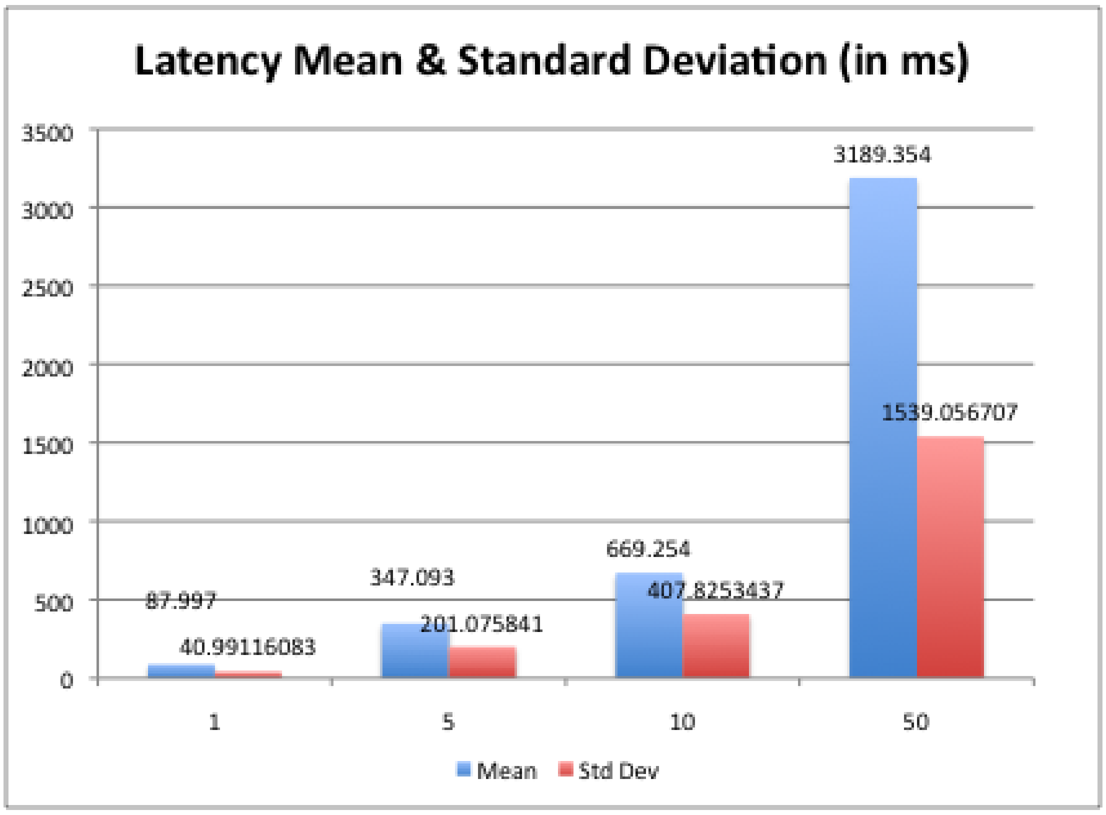

Figure 8: Latency Mean & Standard Deviation

To show scalability, we have calculated the "normalised" mean and standard deviation of latency figures for 1, 5, 10 and 50 concurrent clients. The mean latency per client decreases steadily as we add clients. The normalised standard deviation / client decreases very slowly.

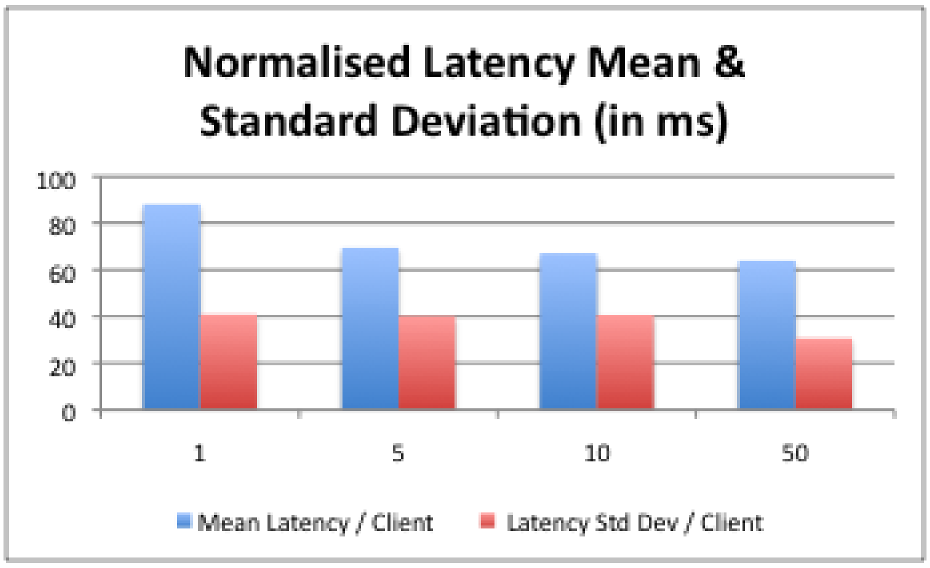

Figure 9: Normalized Latency Mean & Standard Deviation

The performance profile for each set of experiments can be viewed by showing the probability distribution for experiment with varying numbers of clients. These are showing in the following charts.

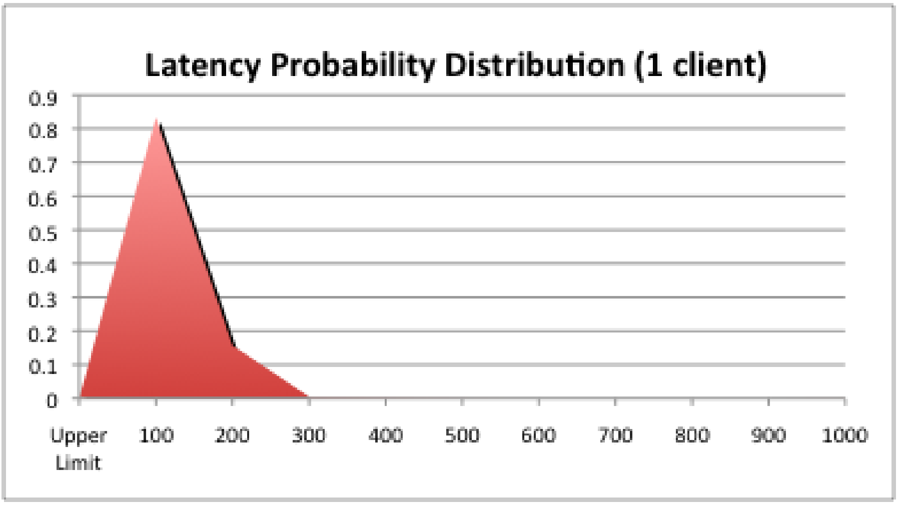

Figure 10: Latency Probability Distribution (1 client)

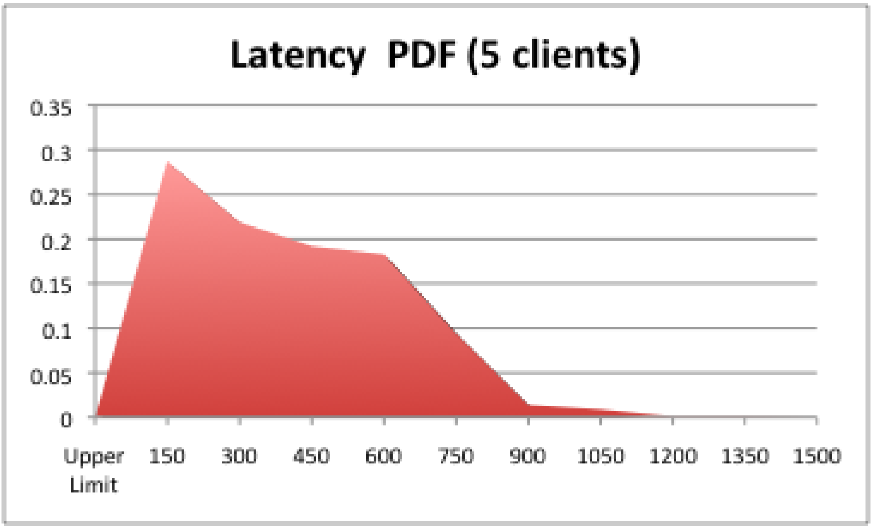

Figure 11: Latency PDF (5 clients)

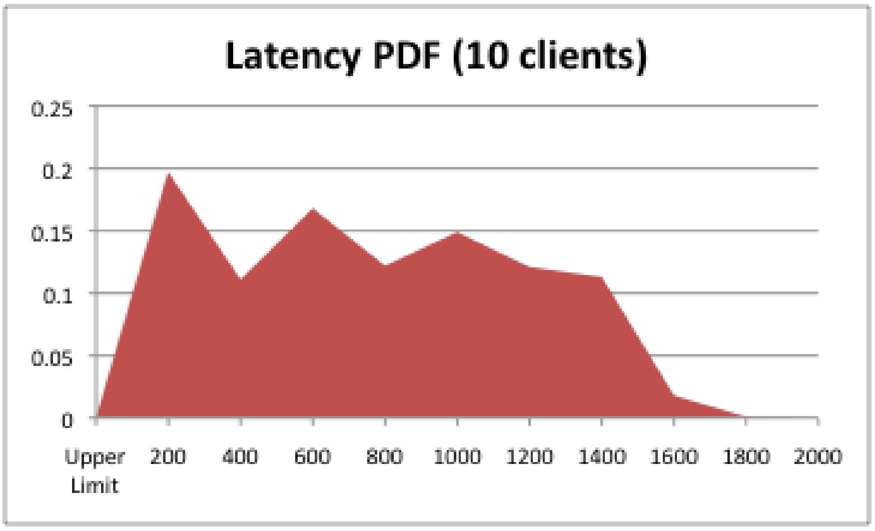

Figure 12: Latency PDF (10 clients)

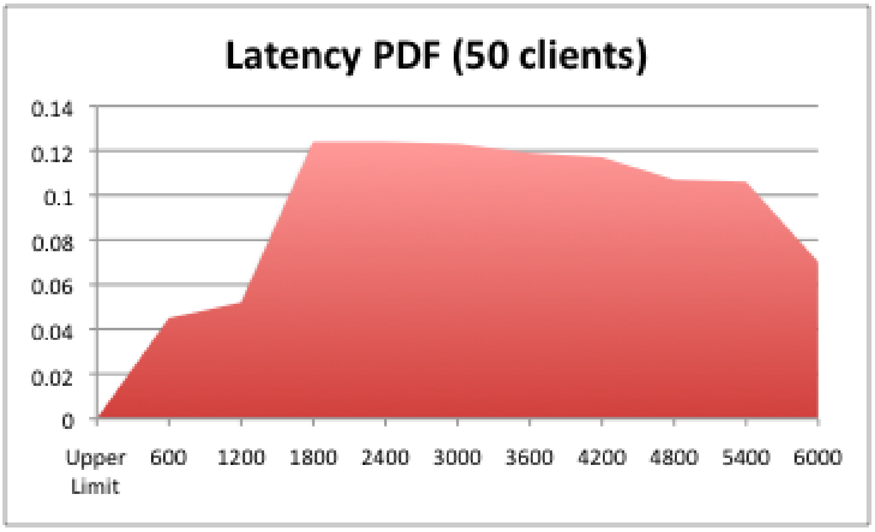

Figure 13: Latency PDF (50 clients)

2.1.2.2 Result analysis

The results analysis can be subdivided into functionality, reliability, performance and scalability.

Functionality

As the error rate was 0% throughout the set of experiments, we deem that the system functions correctly. The expected licensing result was returned in all cases where a response was received. No "race conditions" or other errors normally associated with incorrect use of database transactions were observed. The requests for each one of a set of 10 content items were randomly generated during the experiments to ensure that such errors would appear should they exist but none were observed.

Reliability

We observed that in 1 of the 4000 test cases executed the system did not respond due to a Java networking exception. Upon further analysis this was caused by a network timeout within the Sailfin J2EE Application Server. As the default handling of such an exception is to retry the HTTP socket connection we do not believe this is a serious problem.

Generally, considering the 0% error rate and the 0.1% unavailability rate observed during one set of experiments (0.025% overall), we believe we have demonstrated that the system performs reliably while maintaining connections and state for 50 concurrent clients.

Performance

The system was tested with 50 clients, generating a request every 73.56 milliseconds, a rate of ~13.6 requests / second / client. The objective of the test was to see how stably the system performed under load. It is acknowledged that the System Under Test is unsuitable for a network operator deployment where such a system could easily be deployed on a cluster with a multiple of the aggregate power. For example we are aware from our work with the NGN Test Centre[1] in WIT that Ericsson use 8 full SUN x6250 blades with Xeon Quad-Core 2.5 GHz processors connected using the SUN Blade 6000 Chassis for a "typical" app server deployment, generally configured with N+1 redundancy. This is more than 64 times the power of the System Under Test. Therefore the performance of our system is a less important consideration than it's scalability.

Scalability

However, we have observed that our system, even in its limited configuration, scales in a predictable manner. Latency mean and standard deviations, normalised by dividing by the number of concurrent clients, are consistent as we scale from 1 to 50 clients. The steady decrease in both figures demonstrates that the system is performing more efficiently under heavier load. On average, it's taking less time to respond to each client.

We attribute the multiple peaks of the Latency Probability Distribution Functions (PDF's) to the effects of the HTTP load-balancing function within the Sailfin Application Server. As the load increases, with the number of concurrent clients, the application server attempts to balance the requests to each servlet corresponding to an open socket, peaking at 50 for the final tests. Hence, the 50 client PDF is "flatter" around the mean.

2.1.3 Content indexing CI-SA

2.1.3.1 Result presentation

The CI-SA has been evaluated with respect to the time required for serving various user requests. More specifically, the following requests were analyzed:

- Content Publication
- Query
- Content Selection

[1] NGN Test Centre, http:/www.ngntestcentre.com

All the requests arriving at the CI-SA are buffered and processed in a first-in-first-out mode. This means that no overhead is introduced from parallel request processing and consequently the overall service delay is only affected by the length of the requests' queue.

The methodology for collecting the measurements based on the use of adaptations of the BCT client that were tuned to generate multiple messages towards the CI-SA. All the steps in the processing of a request were logged at the CI-SA and timestamped. The logs were analyzed and the following results were collected:

Processing Task	Time (milliseconds)
Message Parsing	2
Completed Publication Processing (db update, message generation)	38
Create new Stream ID and add first Peer to OM	18
Total Time Processing Publication	61

Table 4: CI-SA Publication Request Processing

Processing Task	Time (milliseconds)
Message Parsing	1
Completed Query Processing (db query, message generation)	45
Total Time Processing Query	50

Table 5: CI-SA Query Request Processing

Processing Task	Time (milliseconds)
Message Parsing	2
Adding new Peer to OM for selected Stream ID – Get Neighbours	23
Refresh XML db entry with new peer and create response message	77
Total Time Processing Publication	102

Table 6: CI-SA Content Selection Request Processing

2.1.3.2 Result analysis

The collected measurements indicate that CI-SA spends from 50 to 100 milliseconds per request. This time period is not the one perceived by the user since there are more delays introduced by the transfer of the message among the IMS components. Although the time required for message processing in the CI-SA is not so long, there may be need for the introduction of parallel processing mechanisms in case the CI-SA is deployed in a setup that involves thousands of users.

2.1.4 Overlay Management OM-SA

2.1.4.1 Result presentation

The scope of this experiment was to simulate the phenomenon of “churn”, which can lead to an unbalanced p2p network and cause problems to the proper operation of scheduling algorithms. In order to simulate this phenomenon, we used 70 Virtual Machines, where one p2pEngine was launched in each VM. We applied a uniform function for the insertion of the clients in the overlay, which phase lasted 70 seconds. The overlay didn’t change between 70 and 160 seconds, but from 160 to 200 seconds we caused 40 departures of clients. The last phase lasted until 260 seconds, where the transmission of the video stopped.

In order to evaluate the execution of the scheduler and the behavior of the whole system in arrivals and departures, we measured the successfully played blocks in four time slots (60, 110, 180 and 250 seconds). At 60 seconds the insertion phase was in progress, at 110 all the clients had arrived, at 180 seconds the departing phase was executed and at 250 seconds the system was balanced and the transmission reached the end.

In the following graphs (figures 14, 15, 16 ,17), the successfully played blocks are presented. We can remark that the CDF is slightly changed among the four time slots.

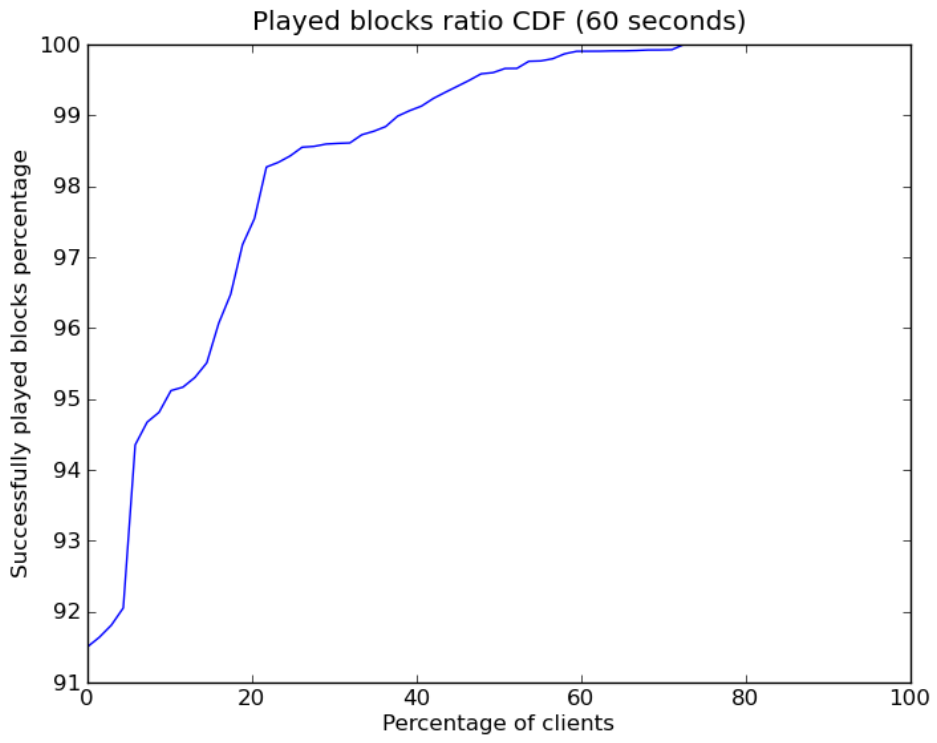

Figure 14: Played blocks ratio (60 sec)

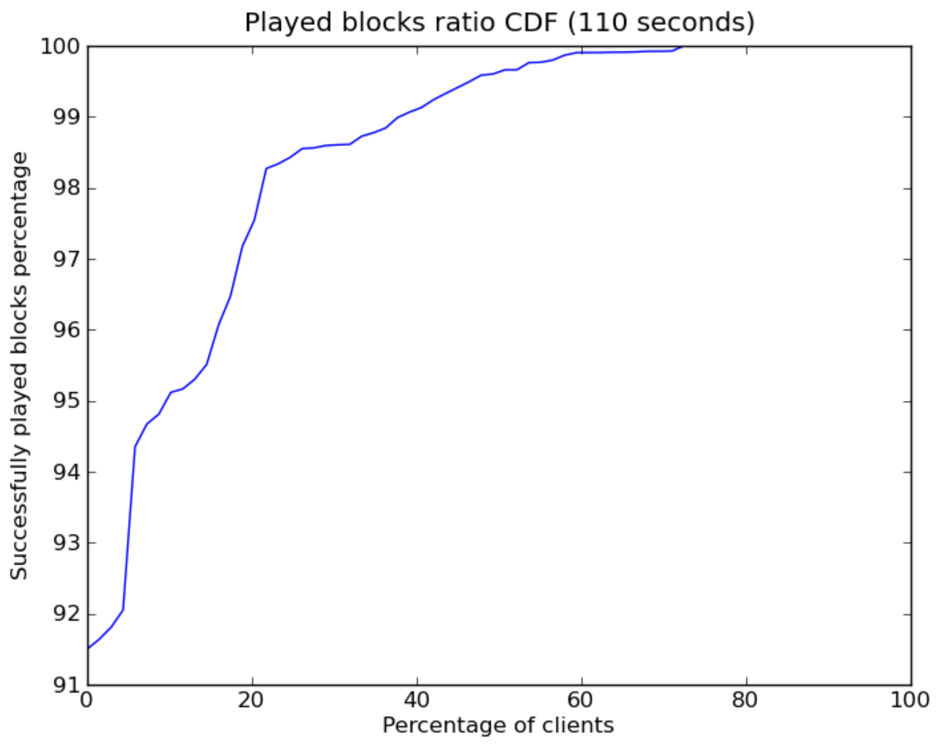

Figure 15: Played blocks ratio (110 sec)

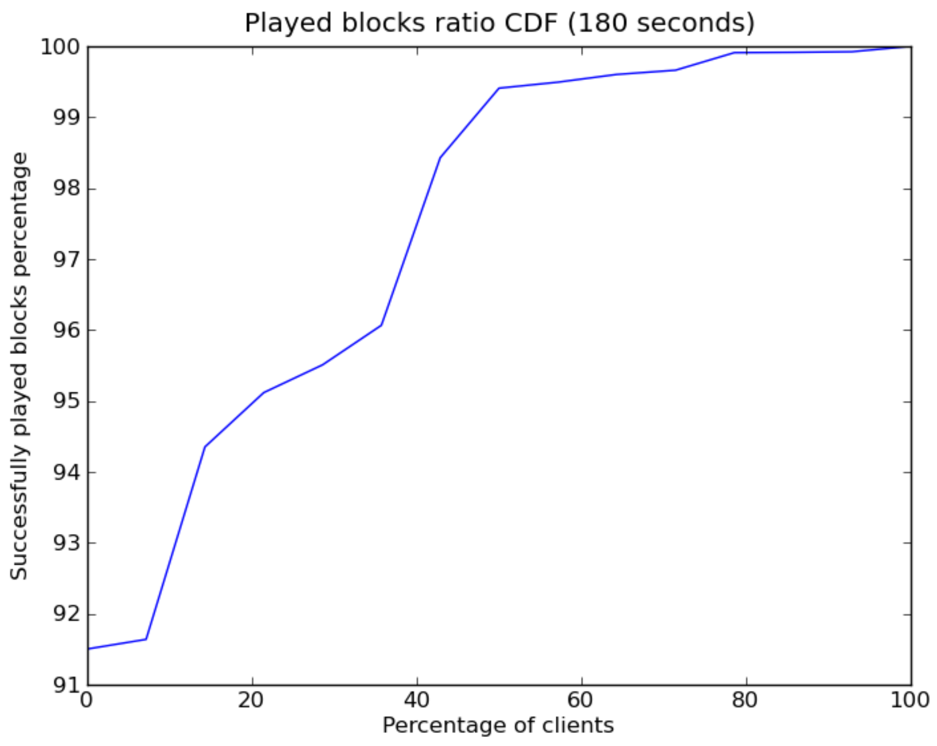

Figure 16: Played blocks ratio (180 sec)

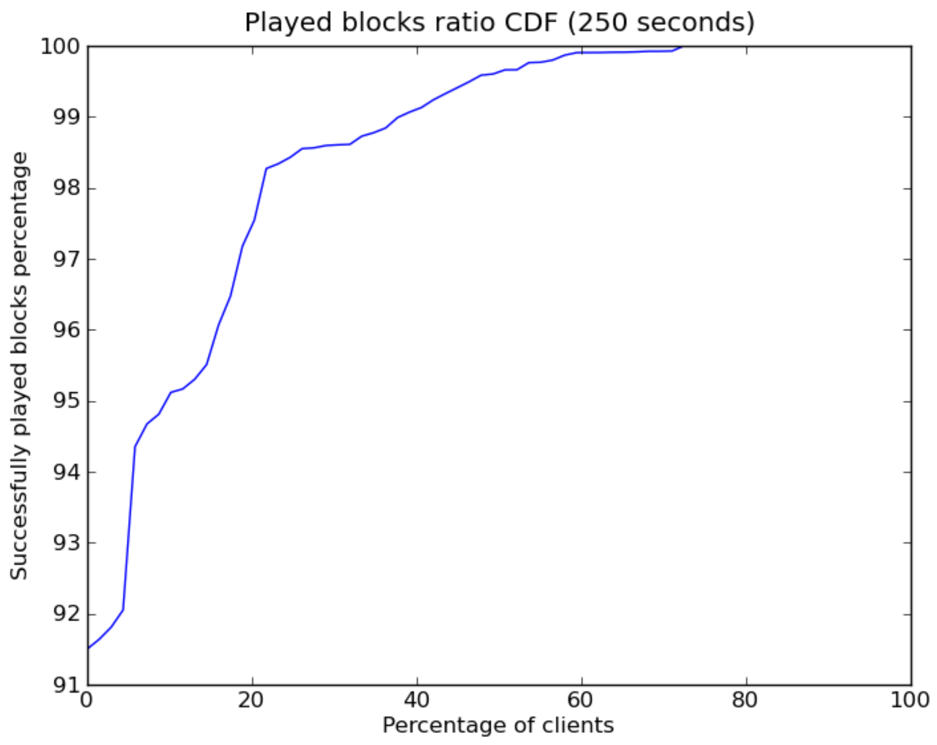

Figure 17: Played blocks ratio (250 sec)

2.1.4.2 Result analysis

The system reacts effectively to client insertions and departures, which fact declares the proper operation of the overlay management algorithm. The algorithm can adjust the overlay formation in a way that does not affect the transmission of the blocks in the running clients. Moreover, the percentage of the successfully played blocks is more than 95% in all the time slots and for more than 90% of the clients.

Consequently, we reached the conclusion that both the scheduling and overlay management algorithms are tolerant to dynamic peer arrivals and departures.

2.1.5 Transcoding service

2.1.5.1 Result presentation

TID's Transcoding service was tested by using 2 separate sets of tests, one to determine the performance of the transcoder service as the file size of the query increases, and the other as the amount of concurrent queries increases.

It is important to realise that *file size* stands for duration in seconds of the original files to be transcoded, i.e. and that only audio files in .wav format (44100 bits/s, 16 bits per sample, stereo) will be used for the tests.

The test cases were defined and described in D5.2 as:

- Transcoding Performance vs. file size
- Transcoding Performance vs. query's load

A summary of the tests is presented below.

Transcoding Performance vs. file size

A request is made to the transcoder service in order to convert an audio file of N seconds of duration into the five available output formats. The total completion time for the operation is recorded. Files with increasing sizes are processed and the transcoding time is measured.

Logically the total completion time for the operation increases with the file size.

The purpose of the tests is to determine whether or not, the service could be used for real time/on line transcoding of audio and video contents.

Transcoding Performance vs. query's load

A number of parallel queries request the transcoder service and the total completion time is recorded. The amount of concurrent transcoding requests is increased and the completion times are measured.

The total completion time increases with the number of parallel queries.

The purpose of the tests is to determine whether or not, the service could be used for real time/on line transcoding of audio and video contents.

System under Test

The System under Test consisted of one virtual machine installed on a DELL Poweredge 1900, running under VMware ESX Server 3.5 and administing using VMware Infrastructure Client 2.5.0 & VMware VirtualCenter 2.5.0.

	Trasncoding Server
Operating System	Linux openSUSE 10.2 (i586)
Kernel	Linux 2.6.18.2-34-default
CPU	4 x Intel(R) Xeon(R) CPU E5335 @ 2.00GHz
Load Average	0.53 0.32 0.26
Disk	/dev/sda: 44.5 GB Used: 15G Free: 24G /dev/sdb: 34.3 GB Used: 15G Free: 16G
Memory	Total 2GB, Used 1'34GB, Free 0'63GB

Testing procedure

The testing procedure was as follows. A specialized java program was developed which carries out the following tasks:

- Launch the transcoder query, specifying the URL where the original .wav file can be downloaded and an unique identifier to be used in further requests.

Poll the transcoder server with the unique identifier mentioned above in order to find out when the transcoding operation has been completed. A transcoding transaction is considered complete when the original .wav has been transcoded in the five final formats and an URL for each made available for downloading. The format produced are the followings: AAC @32Khz 8kbps, AAC@48Khz 64kbps, AAC@48Khz 128kbps, MP3@24Khz 64kbps, MP3@44.1Khz 128kbps, MP3@48Khz 192kbps

- The Poll interval is configurable and was set as 200ms, in order to avoid too many concurrent requests at the same time.

The total time being measured is the one lapsing from the beginning of the transcoder query to the end of the successful poll. Roundtrip times are therefore being included in order to emulate a real case, when an external agent requests the transcoding of a content and has to wait for the confirmation of the operation, before making use of the transcoded contents.

Transcoding Performance vs. file size

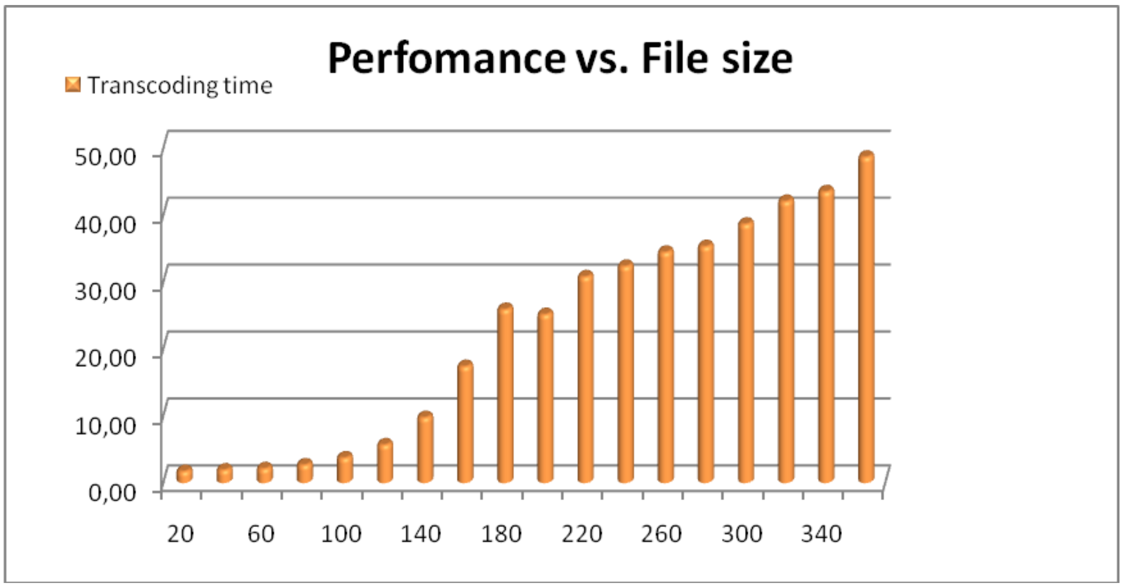

Figure 18: Performance vs. File size

Transcoding Performance vs. query's load

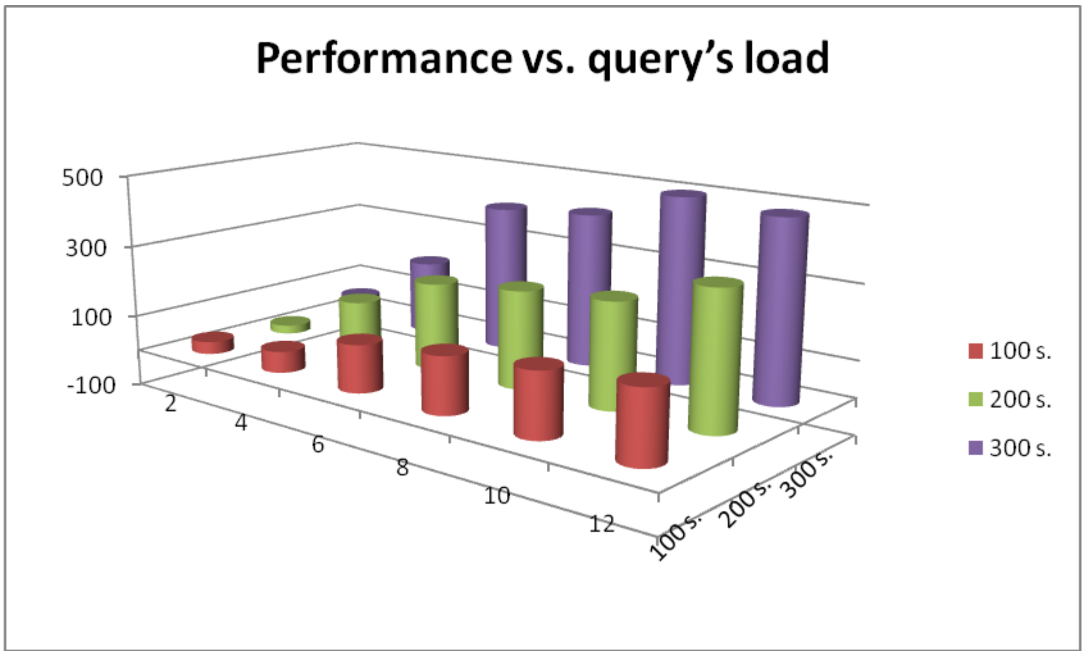

Figure 19: Transcoding Performance vs. query's load

2.1.5.2 Result analysis

As the results show, the transcoding service can only be used for off-line operations or constraint to services that do not require too many parallel connections and/or manage small file sizes.

2.1.6 DVB-T environment

The creation of a DVB-T environment that makes use part of the VITAL++ architecture is presented in this subchapter. By deploying VITAL++ P2P technology, we are creating a VITAL++ DVB-T Environment were the end users are able to use P2P overlays for exploiting the available network Recourses of their "neighbours". Performance evaluation experiments carried-out under real transmission/reception conditions verified the validity of the proposed architecture, besides outlining fields for future research.

2.1.6.1 Overall architecture of a regenerative DVB-T platform

In this context, an overall existing architecture of a DVB-T converged environment is depicted in figure 20. It consists of two core subsystems: a) a central broadcasting point (DVB-T core network), and b) a number of Cell Main Nodes (CMNs) distributed within the broadcasting area.

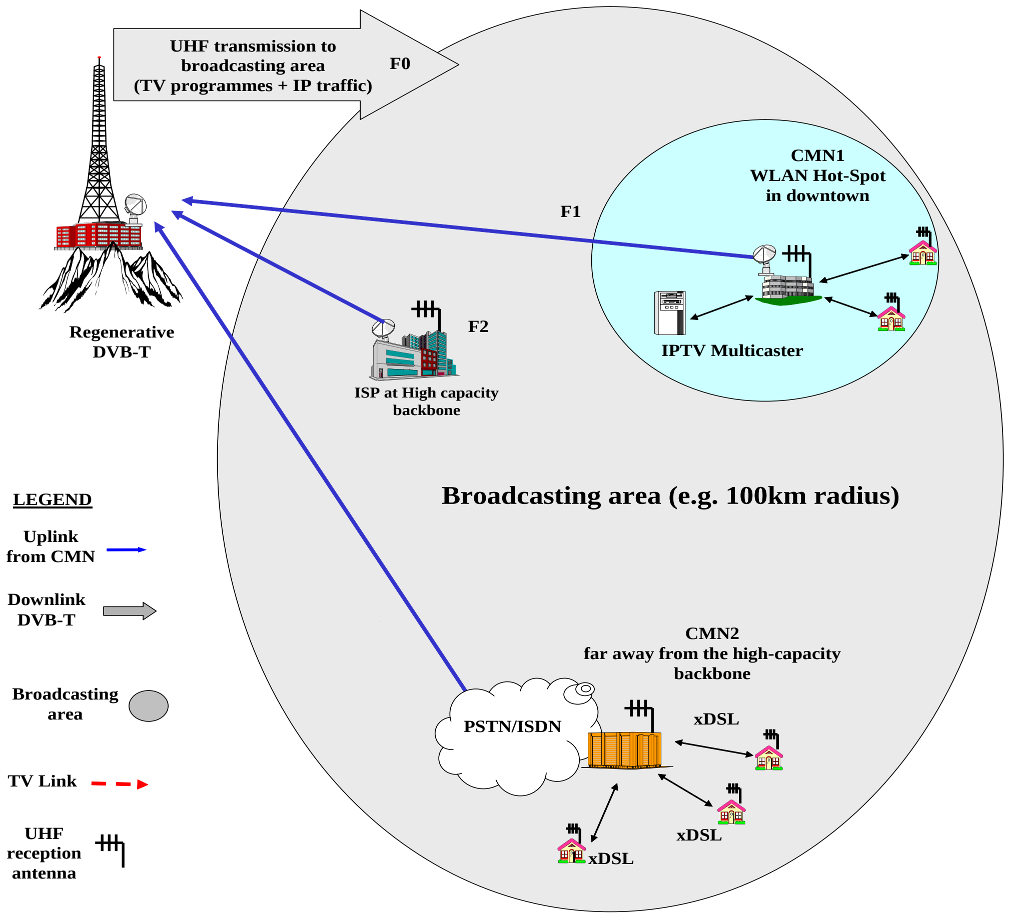
UHF transmission to broadcasting area (TV programmes + IP traffic)
F0
Regenerative DVB-T
F1
CMN1 WLAN Hot-Spot in downtown
IPTV Multicaster
F2
ISP at High capacity backbone
Broadcasting area (e.g. 100km radius)
LEGEND
Uplink from CMN
Downlink DVB-T
Broadcasting area
TV Link
UHF reception antenna
CMN2 far away from the high-capacity backbone
PSTN/ISDN
xDSL
xDSL
xDSL

Figure 20: A DVB/IP environment.

Each CMN enables a number of users/citizens (geographically neighbouring the specific CMN) to access IP services (see figure 21) that are hosted by the entire infrastructure.

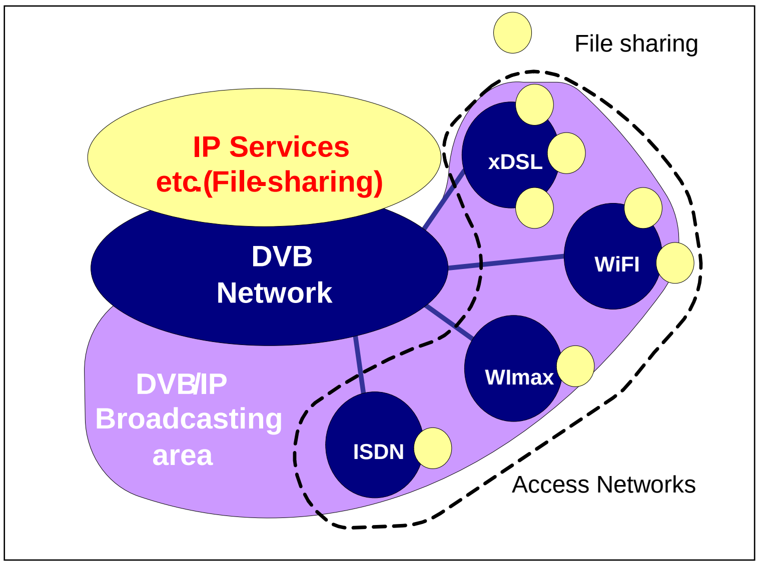

Figure 21: Layered Structure of A DVB/IP Network

The communication between the users and the corresponding CMN (access network) is achieved via broadband point-to-multipoint links (i.e. WLAN, WiFi, ADSL, etc.). Each CMN gathers all IP traffic stemming from its own users (i.e. file sharing data) and forwards it to the central broadcasting point via dedicated point-to-point uplinks. IP traffic stemming from all CMNs is received by the broadcasting point, where a process unit filters, regenerates and multiplexes it into a single transport stream (IP-multiplex) along with the digital TV programme(s) stemming from the TV broadcaster(s) (TV studio), towards forming the final DVB/IP "bouquet". Each user receives the appropriate IP reply signals indirectly via the corresponding CMN, while receiving custom digital TV programmes (e.g. MPEG-2) via the common DVB-T stream. In such configuration, both reverse and forward IP data traffic are encapsulated into the common DVB-T stream**,** thus improving the flexibility and performance of the networking infrastructure. Furthermore, the cellular conception that is adopted utilises the DVB-T stream in a backbone topology, which interconnects all cells that are located within the broadcasting area. Thus, a unique virtual common IP backbone is created, which is present at every cell via its CMN.

2.1.6.2 P2P Overlay in a DVB-T Architecture

Following the design approach discussed previously, this section presents the overall configuration of a DVB-T system that utilizes the VITAL++ P2P Overlay (see figure below). By deploying VITAL++ P2P technology, we are creating a VITAL++ DVB-T Environment were the end user are able to fully utilize the network infrastructure by exploiting the available network resources of their "neighbours".

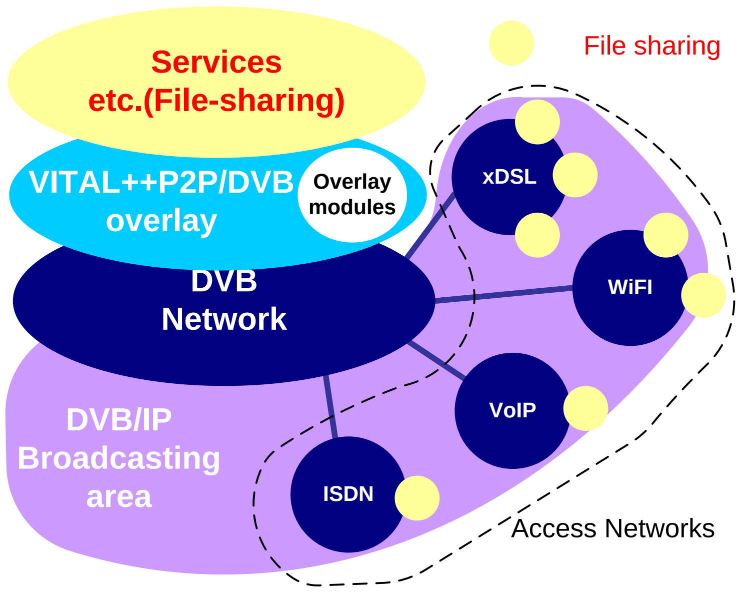

Figure 22: DVB-T with P2P Overlay

The overall architecture of a DVB-T system that utilizes the VITAL++ P2P is presented in figure 23. In such a scheme, every Cell Main Node (CMN) of the DVB-T network is considered as an independent network that utilizes P2P users in a DVB-T network.

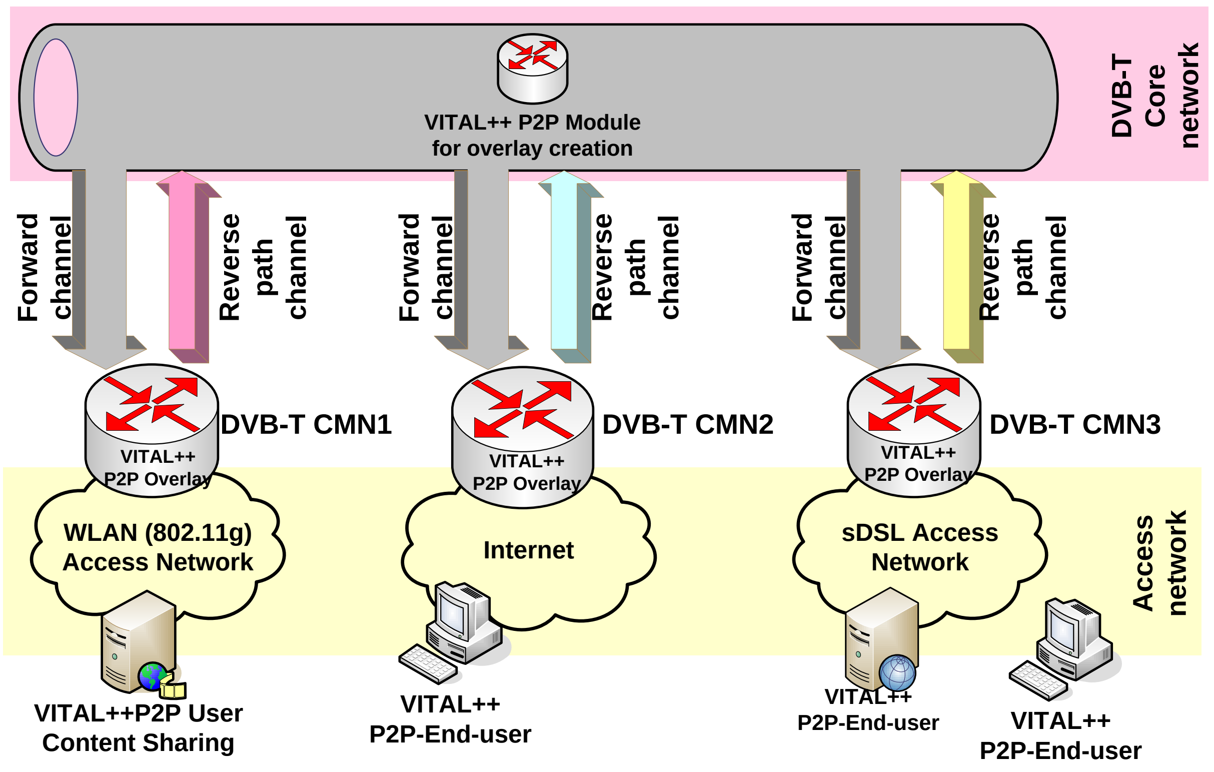

Figure 23: Overall architecture of a VITAL++P2P DVB-T infrastructure.

The VITAL++ users connect to the DVB-T network through a P2P aware CMN, which monitors and routes the hosted users. Every CMN communicate with the other CMNs that exist in the DVB-T broadcast region via the DVB-T core network making use of the reverse path (point-to-point uplink). All the traffic stemming from the VITAL++ end users is passing through the CMN where the traffic is forwarded towards the central broadcasting point (UHF transmission point). At the central broadcasting point, all the received traffic (Services/Users data and signalling information of the P2P protocol) is encapsulated and transmitted in the whole broadcasting area. The VITAL++ users are receiving information from a VITAL++ module for overlay creation that is located in the central broadcasting point. In this way, every P2P-end user is able to learn information about the location of the services, their location, the completion time, and the location of other P2P-end users.

2.1.6.3 Experimental validation of the proposed DVB/P2P enable infrastructure

Following the design specifications that were presented previously, this section elaborates on the implementation of a prototype platform conforming to the proposed architecture, which serves as a test-bed for conducting performance evaluation tests under real transmission/reception conditions. The overall configuration of this experimental test-bed (as depicted in the figure below) comprises of:

- A DVB-T platform, where the common DVB-T stream is transmitted in channel 40 of the UHF band (i.e. 622-630MHz), utilizing 8K operation mode with 16QAM modulation scheme, 7/8 code rate, 1/32 guard interval and the multi-protocol encapsulation mechanism (MPE) for the distribution of the IP datagrams. These transmission parameters provide a total available downlink capacity of about 20.5Mb/s, according to the DVB-T standard, part of which (12.5Mb/s) was allocated among three digital TV programs (MPEG-2 live and non-live TV broadcasts), while the rest bandwidth was dedicated to IP services (i.e. 8Mb/s).
- A VITAL++ module for overlay creation that is located in the central broadcasting point, providing initial information's about the location of the services, the completion time, and the location of other P2P-end users.
- A CMN (namely CMN1 in 24) located in an urban area, providing access to file sharing applications (i.e. a 50Mbyte audiovisual data), hosted by the "active user". The communication between this CMN and the regenerative DVB-T platform was via a one-way point-to-point link IEEE 802.11g (uplink), while downlink data are received by the CMN over the DVB-T broadcasting stream. The communication between the active-user and this CMN (access network) is over IEEE 802.11g full-duplex links.
- 4 rural-based CMN (see 24) located 10 kilometres away from the regenerative DVB-T platform. This CMN serves four VITAL++ end users exploiting sDSL technology (downlink1024Kbps/uplink1024Kbps) in the access network, while communicating with the regenerative DVB-T over a common ISDN line in the uplink, and over the broadcasting stream in the downlink.

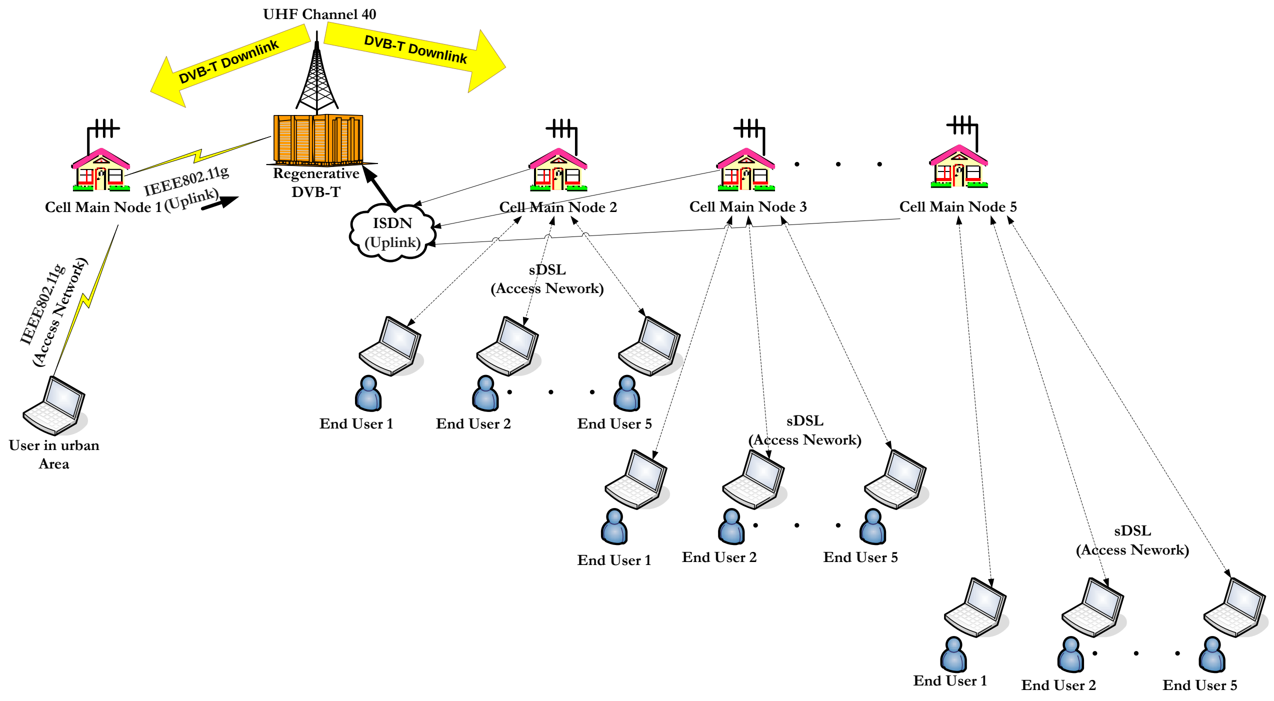

Figure 24: Testbed deployment

Based on this test-bed, a number of experimental scenarios were designed, where the first five sets of experiments were focusing on evaluating the system performance. The seconds sets of experiment is increasing the number of users of the VITAL++ DVB network.

2.1.6.4 Result presentation

Towards validating the overall system's performance and evaluating its capability, a sets of experiments were designed in order to evaluate the system performance. In these experimental scenarios, the users located within the rural area CMN's (CMN2, CMN3, CMN4, and CMN5) were requesting a streaming file hosted by the active-user located within CMN1.

More specifically, when the 5 users were simultaneously requesting the streaming file, the experimental results indicated that the 8Mb/s DVB-IP channel in the DVB-T downlink was underutilized (about 12%), while the utilisation of each ADSL uplink (in the access network) was about 90%. The duration of the experiment was depending on the file size with the smallest one (10 MB) to be downloaded in 84 seconds. The round trip time in these sets of experiments was varying around 107 msec among the urban CMN to the rural area CMNs.

The second sets of experiment validated the whole infrastructure when we increase the number of the end users. More specifically, when 10 users were simultaneously requesting the streaming file, the experimental results indicated that the 8Mb/s DVB-IP channel in the DVB-T downlink was underutilised (about 25%), while the utilisation of each ADSL uplink (in the access network) was about 98%. The duration of the experiment was depending on the file size with the smallest one (10 MB) to be downloaded in 123.4 seconds. The round trip time in these sets of experiments was varying around 107msec among the urban area CMN to the Rural area CMNs.

In the final set of experiment, the scenario is the same but this time we have fifteen users requesting the file. All the other network aspects remain the same. The experimental results indicated that the 8Mb/s DVB-IP channel in the DVB-T downlink was underutilised (about 35%), while the utilisation of each ADSL uplink (in the access network) was about 98%. The duration of the experiment was depending on the file size with the smallest one (10 MB) to be downloaded in 163.4 seconds. The round trip time in these sets of experiments was varying around 107msec among the urban area CMN to the Rural area CMN.

2.1.6.5 Result analysis

The results analysis can be subdivided into operability, reliability and performance.

Operability
As the use of the VITAL++ overlay modules (P2P engine, P2P Server, P2P client) where installed in a DVB-T network environment with no problem, we can come to the conclusion that the VITAL++ P2P modules created from UoP for the aspect of the project are well documented and compatible with the DVB-T test bed that is located in CTRC premises.

Reliability
We observed that when 5 users were requesting the stream, the losses in the DVB-T network were in an acceptable rate of 0-1.5% losses. In this aspect we can come to the conclusion that if a small number o users are requesting the file the VITAL++p2p client is able to provide the file. When we increased the users the losses were raising reaching 6% losses for the 20 users

in order to overcome this problem a new version of VITAL++ client was created that was taking in to account the large one way delay[2] of the DVB-T platform.

Performance

The Performance evaluation experiments carried-out under real transmission/reception conditions verified the validity of the proposed architecture. Enabling, however, dynamic peers' selection prior to the overlay creation as a matter of network characteristics (e.g. BER, Jitter, delay, etc.) and specific service/content quality requirements, constitutes a field for future research, aiming at improving the system performance, scalability and guaranteed QoS.

2.1.7 BCT-Client and derivatives

The BCT Client implementation, integrating also the P2P engine from UoP, has been evaluated with respect to the system resources that requires and more specifically with respect to memory consumption and its relation with the number of overlays joined for the acquisition of content. The measurements collected regard two different instantiations of the BCT client environment:

- The Super-Peer instantiation that instantiated the SoftMix scenario
- The Vital++ end-user application that has been extensively used in the demonstration of the project outcomes

In the first case the collected measurements present the memory demands from the Publisher's perspective, while in the second case there are results regarding the behaviour of the BCT Client with respect to the resources both as a media consumer and as a media producer.

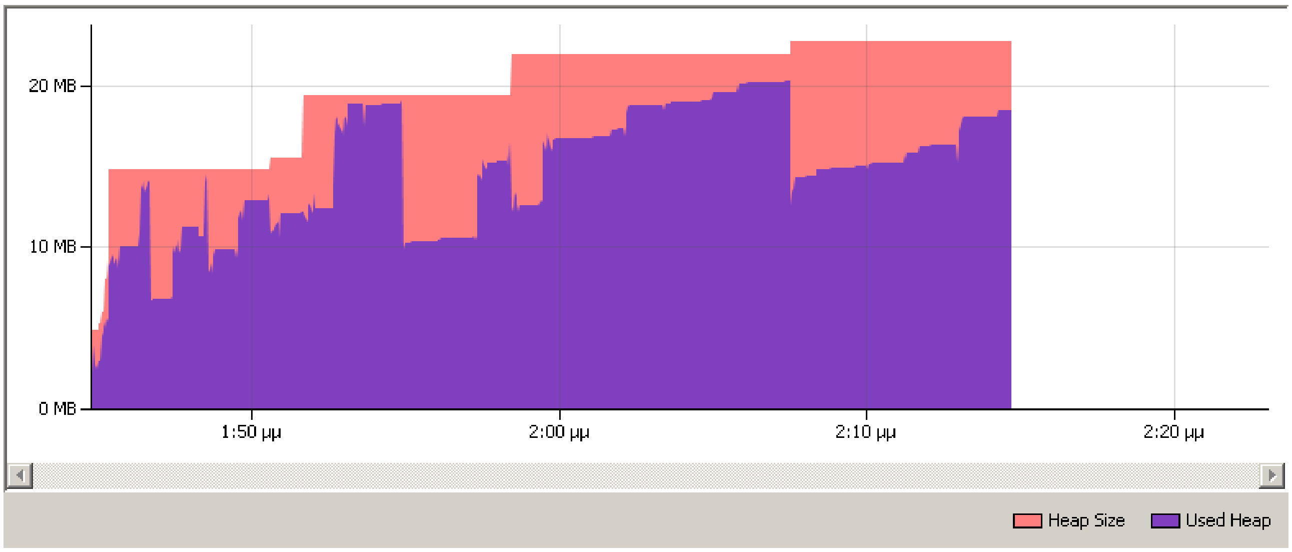

Figure 25: Super Peer Instance publishing 14 items (one every 60 seconds) – memory graph

The first set of results (Figure 25) regards the memory demands of the Super Peer while publishing 14 audio files between 2 and 4 Megabytes. Each publication is following the

[2] [12] G. Almes, S. Kalidindi, M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, IETF September 1999.

previous one after 1 minute. Every time a file is to be published we have an additional memory allocation of a few (4-5) Megabytes that lasts as far as the file is processed and fed into the P2P engine. In many cases the allocation coincides with memory release and thus the overall memory seems not to be constantly increasing. At the end of the process the overall allocated memory remains stable at around 15 Megabytes (the increase observed at the end of the graph is due to the fact that the Super Peer enters a new round of publications).

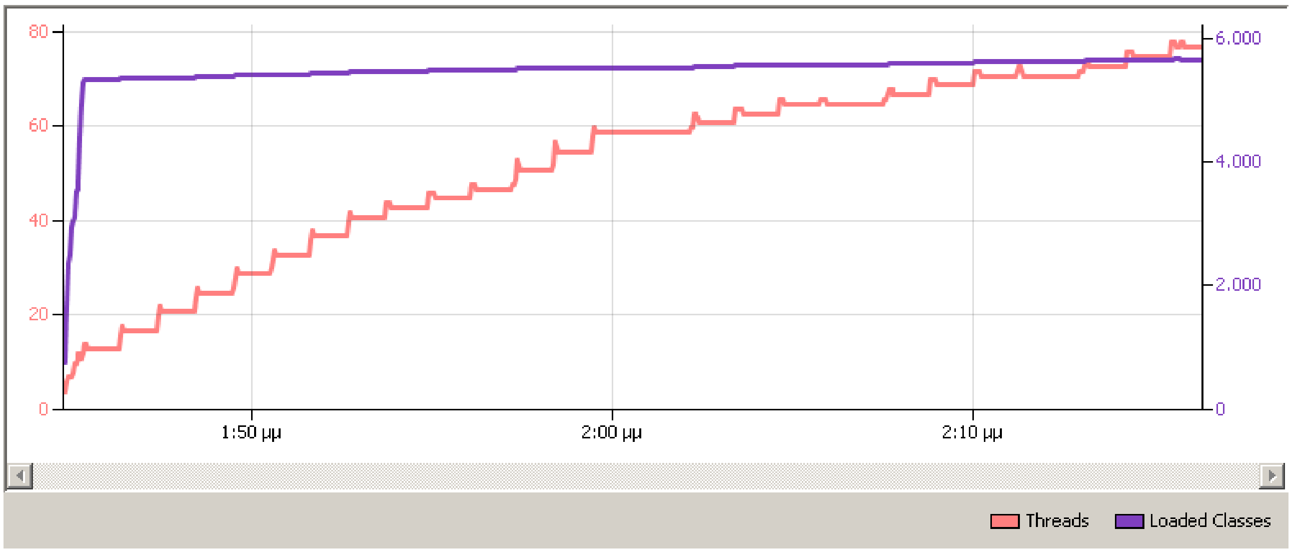

Figure 26: Super Peer Instance publishing 14 items (one every 60 seconds) – thread graph

The corresponding graph that presents the number of started threads is displayed in **Error! Reference source not found.**6.

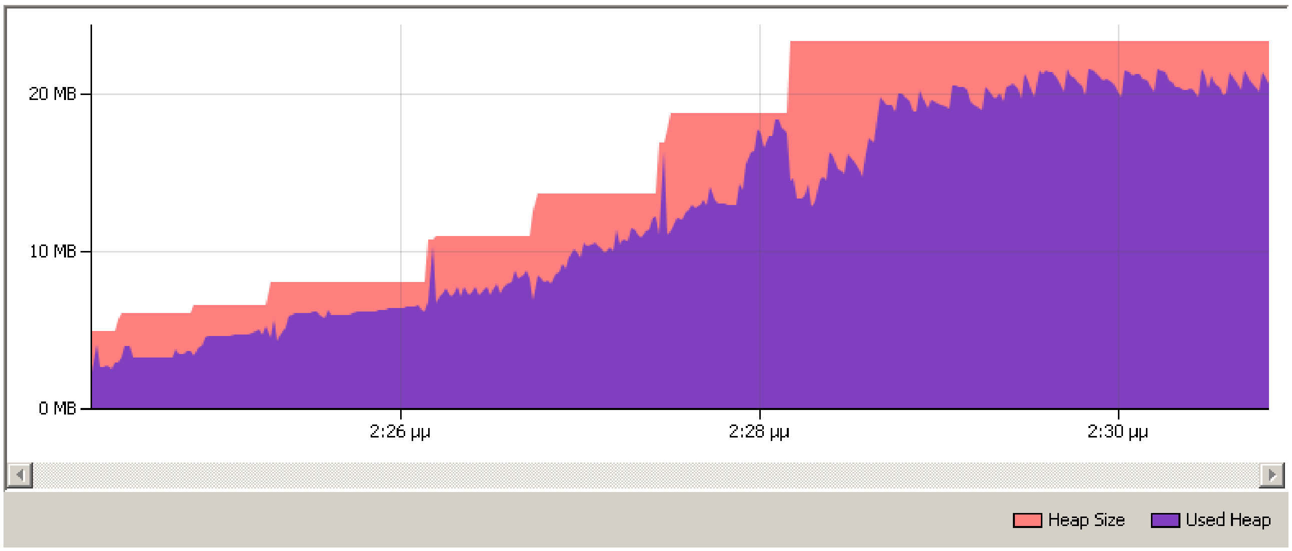

Figure 27: Client Instance consuming 9 items – memory graph

The situation with respect to the memory demands of a media consumer is presented in Figure 27. The request for a new media item is generated manually by the user. We observe an instant allocation of about 3 Megabytes upon initialization of the request that is de-allocated after a while. However, the media playback components that are activated for consuming the media result in a constant increase in the allocated memory. All the items are played in loop mode and this result in the stabilization of the memory demands at the end of the graph.

The corresponding thread generation graph follows (**Error! Reference source not found.**28).

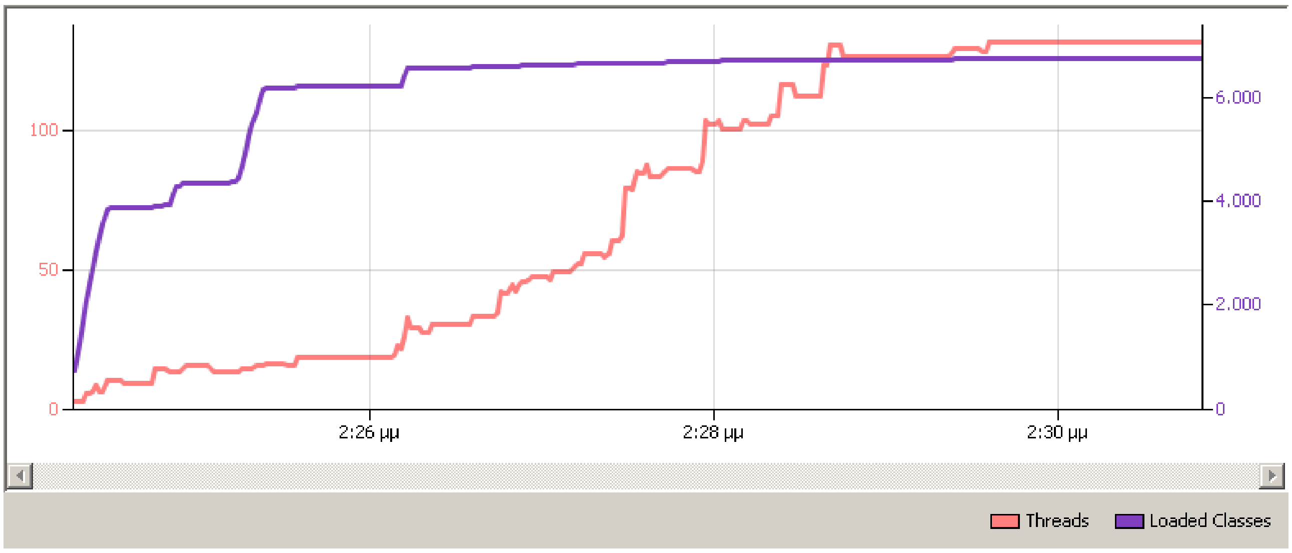

Figure 28: Client Instance consuming 9 items – thread graph

Finally, the memory demands in two additional cases of publication are presented. The first (**Error! Reference source not found.**29) regards the publication of a video file while the second (Figure 30) regards the publication of the user's webcam (live streaming). In both cases the system is stabilized at an allocation of 15 Megabytes. This allocation regards not only media feeding but also media playback.

In the case of video publishing we observe a two step increase in memory since the file is played locally once it has been entirely fed into the P2P engine. This is not the case in webcam publication where engine feeding and stream playback occur simultaneously.

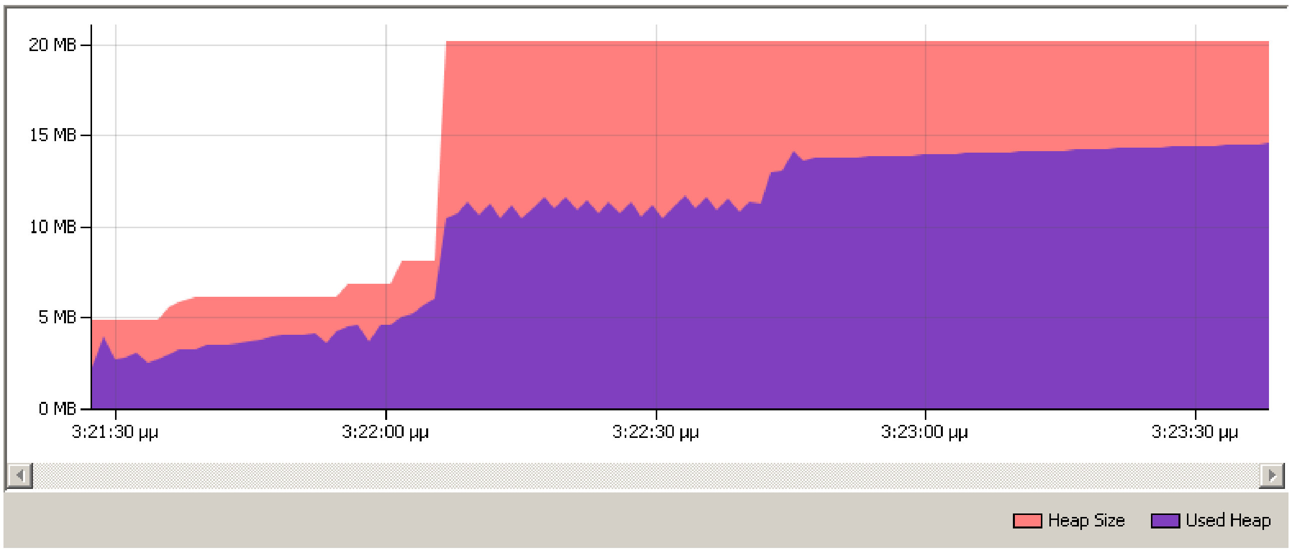

Figure 29: Client Publishing Video – memory graph

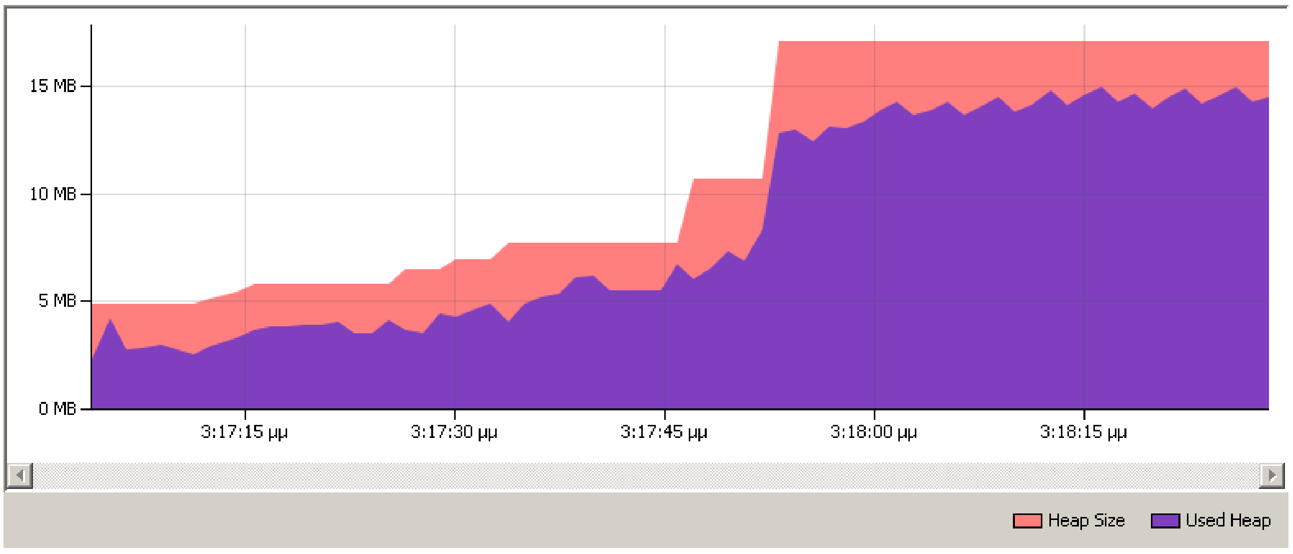

Figure 30: Client Publishing Webcam – memory graph

In all cases apart from the memory demands presented in the above graphs there is a need for 20 Megabytes that are required for the operation of the P2P Engine that is not included in the measurements that produced the graphs since the engine is a separate process.

The profiling tools of the Netbeans IDE were used to produce the graphs above. Memory demands for the P2P Engine were collected from the Operating System monitoring facilities.

3 Real user experiments. Uncertainties and solutions in a real user environment

The real user experiments focused on the evaluation of the performance, scalability and stability of data distribution mechanisms, which consist the core operational mechanism of Vital++ architecture. Specifically, the component that was tested was the P2PEngine which performs the data distribution in both BCT and Monster clients. In order to involve real users of UoP Campus, the P2PEngine was inserted in the UoP P2PClient which is available for downloading to users inside the UoP campus.

The experiments were executed both in a controllable local environment (UoP laboratories) and in a completely dynamic environment, the UoP Campus. As stated in D5.2, the laboratory experiments were performed in three phases, which are described in sections 3.1, 3.2 and 3.3.

3.1 Experiment of linux Virtual Machines

The scope of this experiment was to involve a great amount of clients in the overlay in order to gather statistics related to scalability, bandwidth utilization, flow control, bandwidth waste and scheduler performance.

System Under Test

The System under test consisted of 70 virtual machines installed in 4 XEN Servers. The servers were connected through a Dell PowerConnect 2724 gigabit switch. Every VM hosted one P2PClient (without GUI) and was running Debian squeeze/sid (Unstable) to exploit the more updated python distribution and the new libraries. The bitrate of the transmitted video was 200kbps and the experiment was reproduced for three levels of bandwidth ratio. Specifically the upload bandwidth of every VM was set to 210kbps (ratio=1.05), 220kbps (ratio=1.1) and 230kbps (ratio=1.15). Only the upload bandwidth of the producer of the stream was 1Mbps in order to cover the needs of the "pullscheduler" that is tuned to send the whole stream in two peers. Also the setup time is 3 seconds (buffer size=30 and blocks per second=10).

	XEN Server
Operating System	Debian GNU/Linux 5.0
Kernel	2.6.26-2-xen-686 #1 SMP
CPU	Dual socket quad core Xeon E5405 @ 2.00GHz (8 physical cores total)
Disk	6 250Gb SAS 7200rpm Hard drives
Memory	8GB RAM ECC

The logger server, which was launched in the machine that the P2P Server was running, was responsible for collecting the metrics real-time. The following graphs depict the results of the experiment:

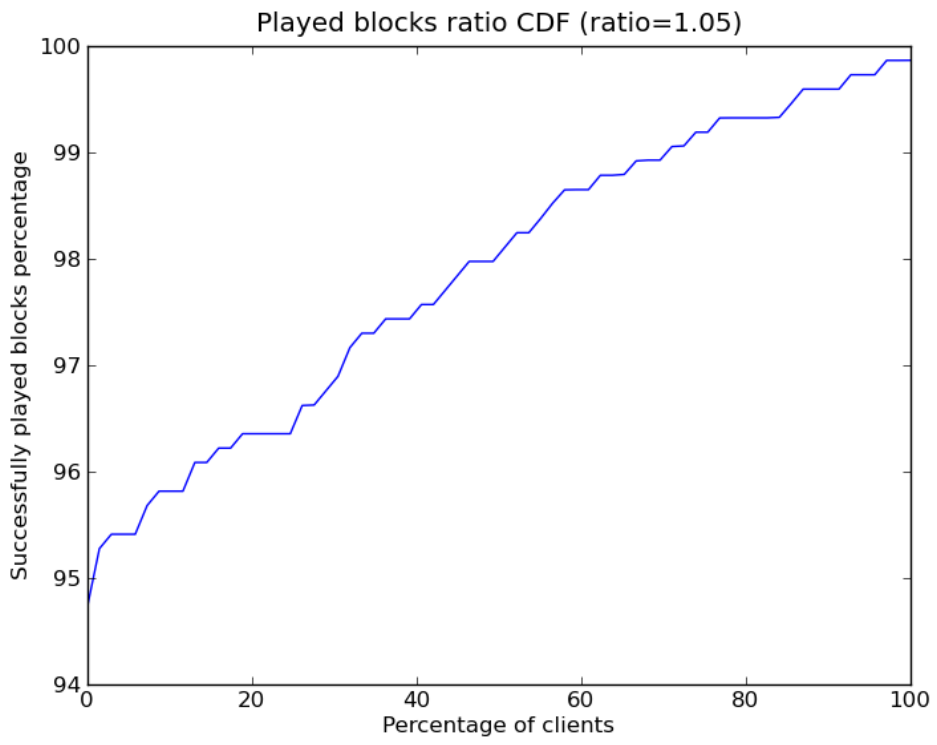

Figure 31: Played blocks ratio (1.05)

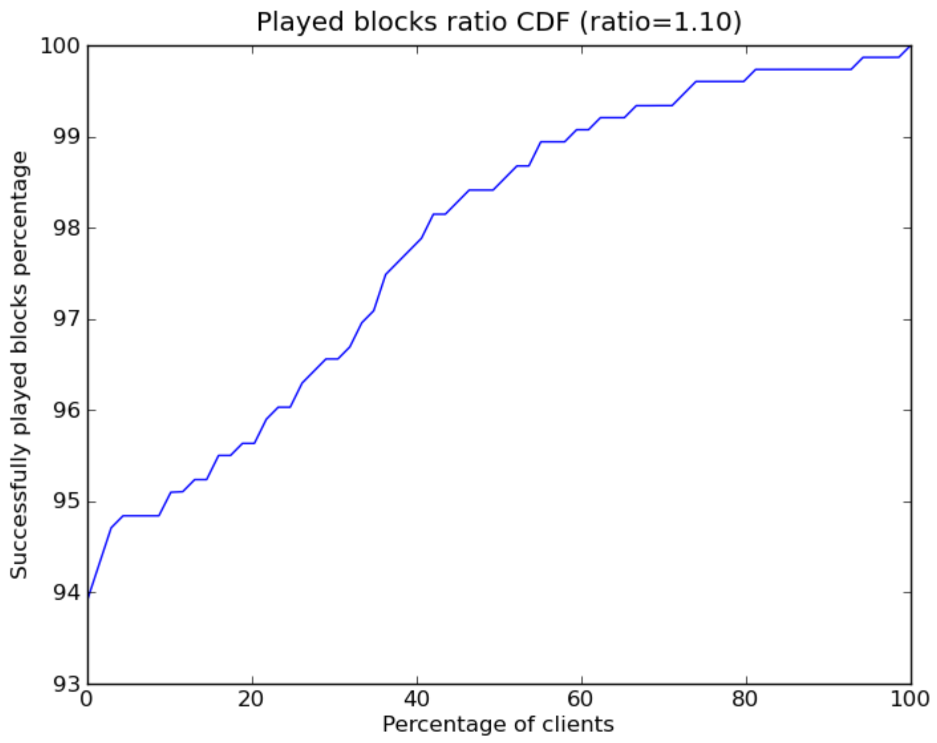

Figure 32: Played blocks ratio (1.10)

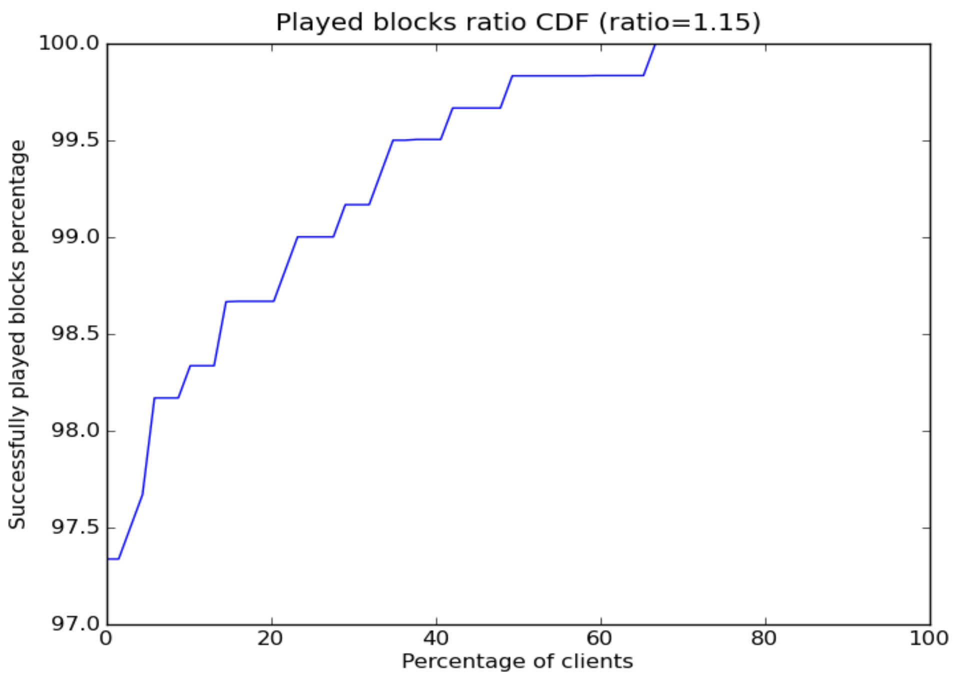

Figure 33: Played blocks ratio (1.15)

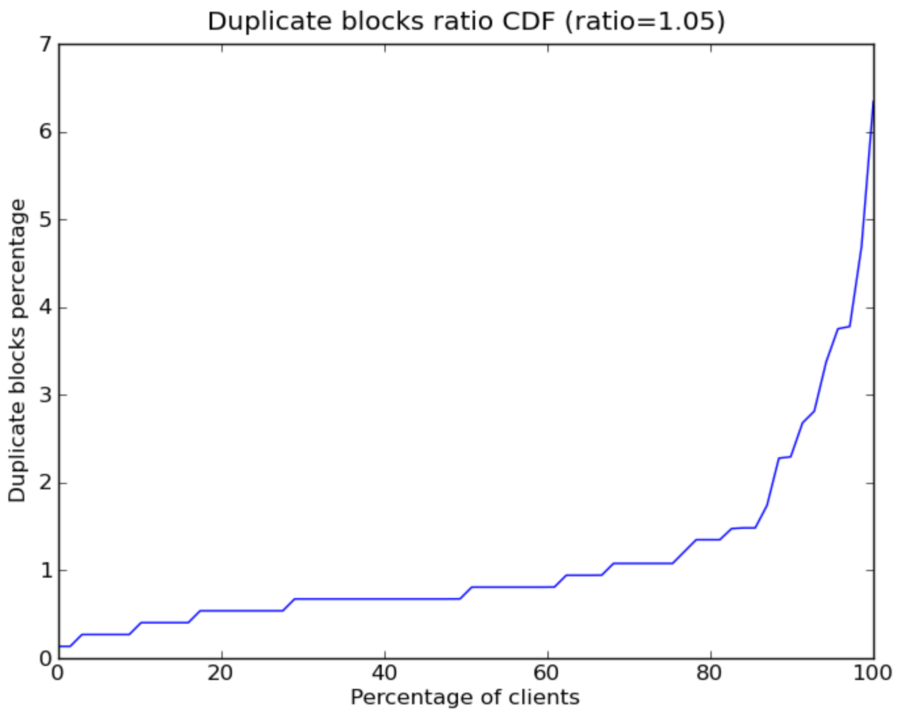

Figure 34: Duplicate blocks ratio (1.05)

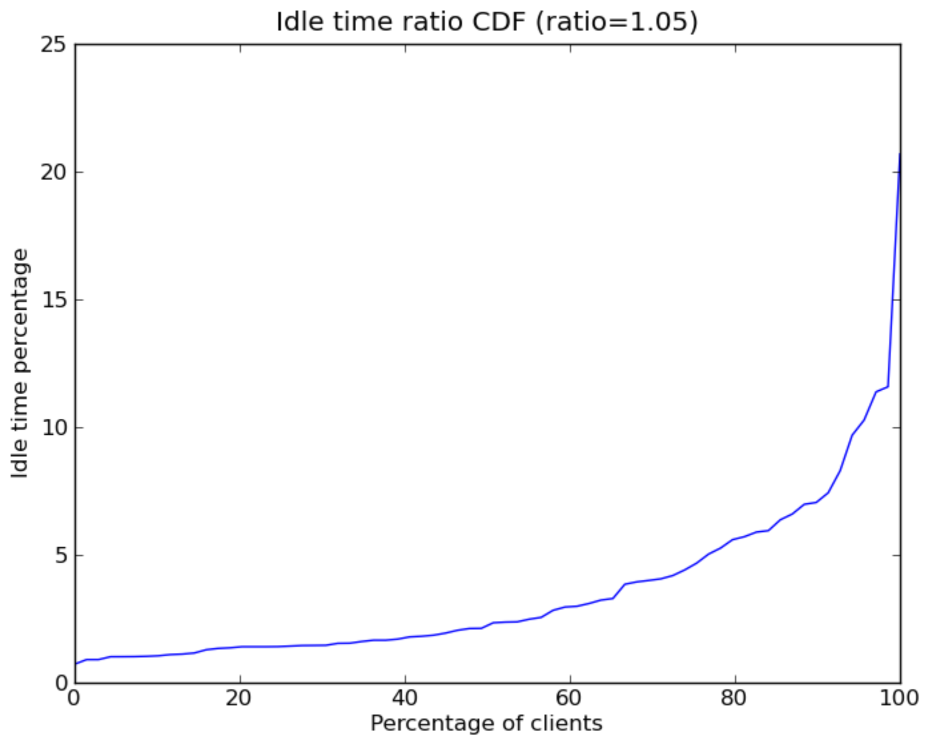

Figure 35: Idle time ratio (1.05)

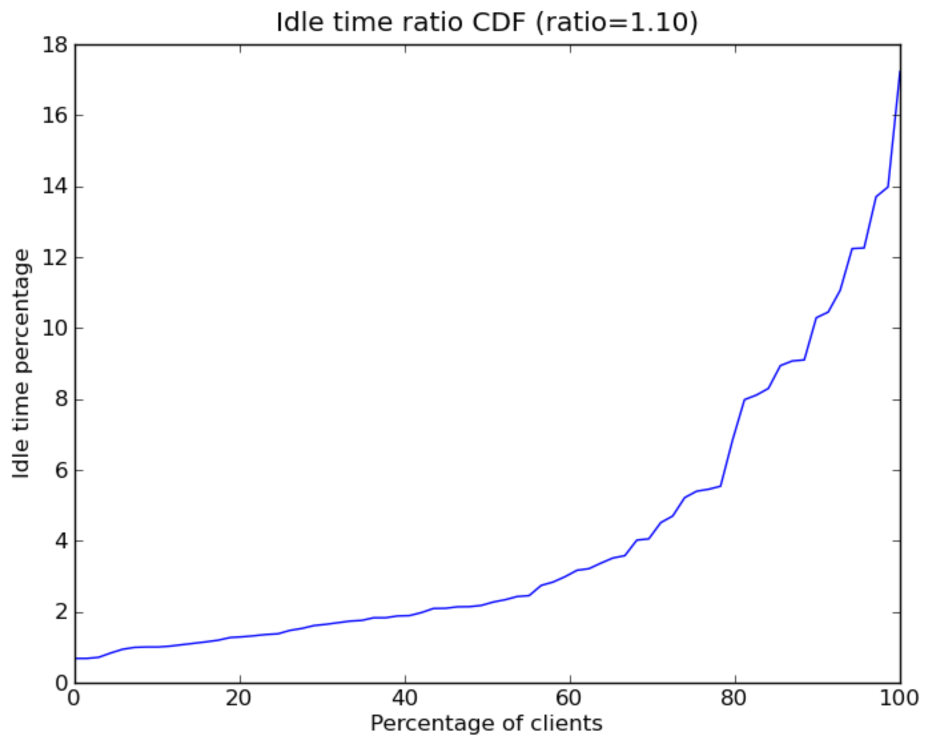

Figure 36: Idle time ratio (1.10)

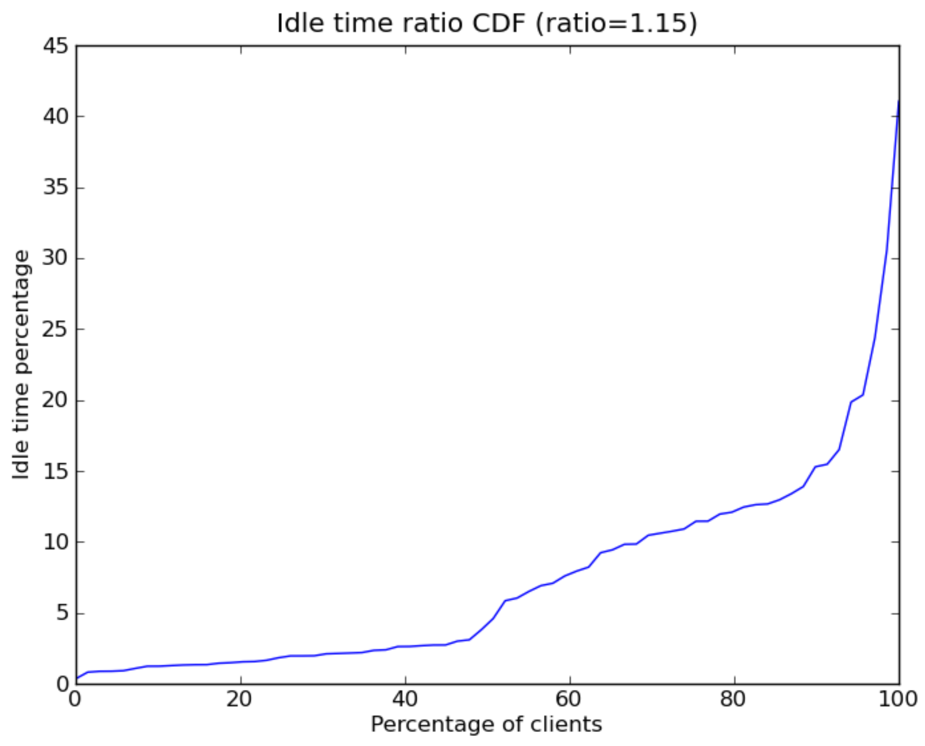

Figure 37: Idle time ratio(1.15)

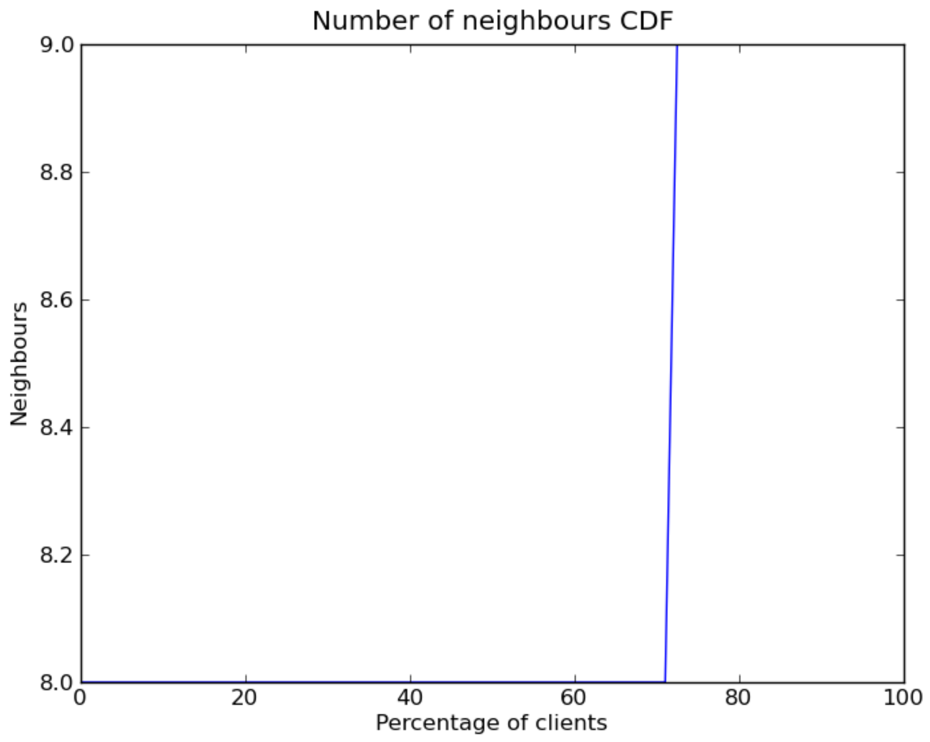

Figure 38: Number of neighbours

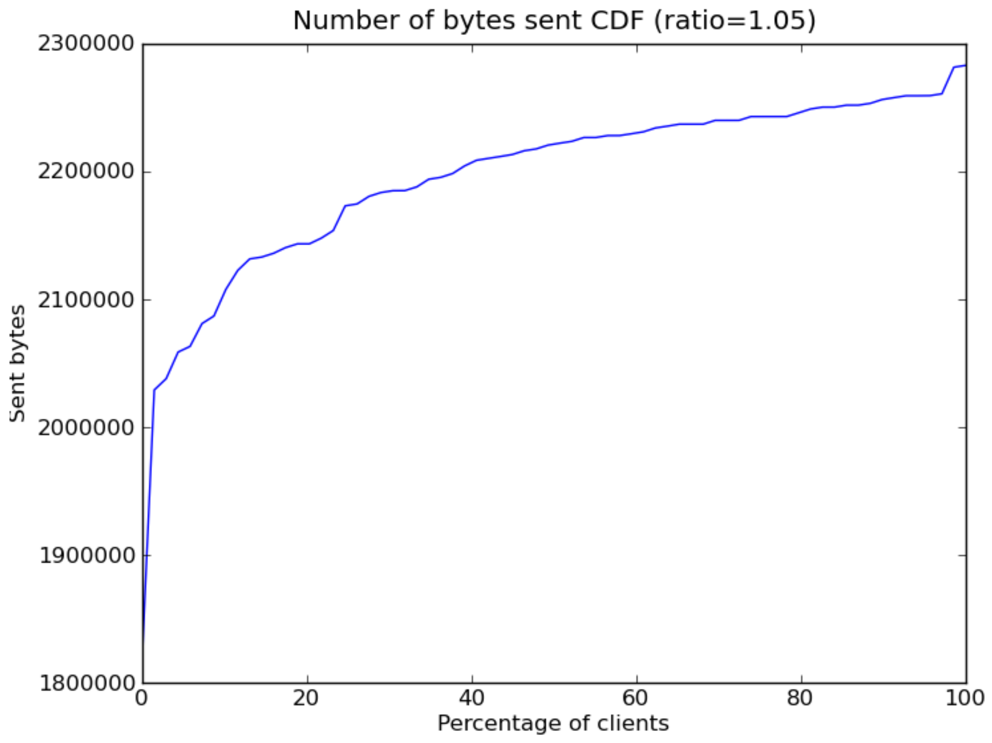

Figure 39: Bytes sent (1.05)

The figures 31, 32 and 33 present the CDF of played blocks of the participating peers. The monitoring of the played blocks parameter can provide estimation about the bandwidth utilization, so we reached the conclusion that the performance of data distribution mechanism in terms of bandwidth utilization is high, even when the average upload bandwidth is close to the video rate (ratio=1.05). Also, the played blocks ratio increases while the average upload bandwidth is getting higher from the video rate. Especially figure 3.3 justifies that 100% of blocks are played in more than 80% of the participating clients.

Idle time and duplicates depict the bandwidth waste and they are presented in figures 34, 35, 36 and 37. The bytes sent figure justifies the performance of the scheduler. Specifically, video is transmitted without losses and duplicate blocks according to figure 34, but the fairness of the algorithm regarding the bytes send from each peer, is not the desired. This fact is justified by figures 35, 36 and 37, because around 25% of the peers remain idle for more than 5 seconds and they don't contribute to the blocks exchange among the neighbours. In simulations, where we evaluate the system performance for higher number of peers (1000-2000), fairness seems to improve as the system scales.

The figure 3.8 shows that every peer has 8 neighbours, so the overlay construction is balanced and the figure 3.9 justifies that the performance of the scheduler is the desired, because even for a low ratio (1.05) the number of sent bytes is high for the majority of the clients.

To sum up, the analysis of the above graphs leads to the conclusion that the results of the execution of the experiment in 70 VMs are very close to the results we have collected during simulations.

3.2 Experiment of windows clients

The experiment of windows clients took place in UoP laboratories and involved 20 P2P Clients running in windows XP machines. The purpose of this experiment was to test the P2PClient in a controllable environment and prepare the application for the real user experiments. The bitrate of the transmitted video was 200kbps, while the upload bandwidth of every client was set to 230kbps (ratio=1.15).

System Under Test

The characteristics of windows PCs that were used during the experiment are presented below.

	PC
Operating System	Windows XP Pro SP2
CPU	Intel Pentium 4
Disk	100GB
Memory	2GB RAM

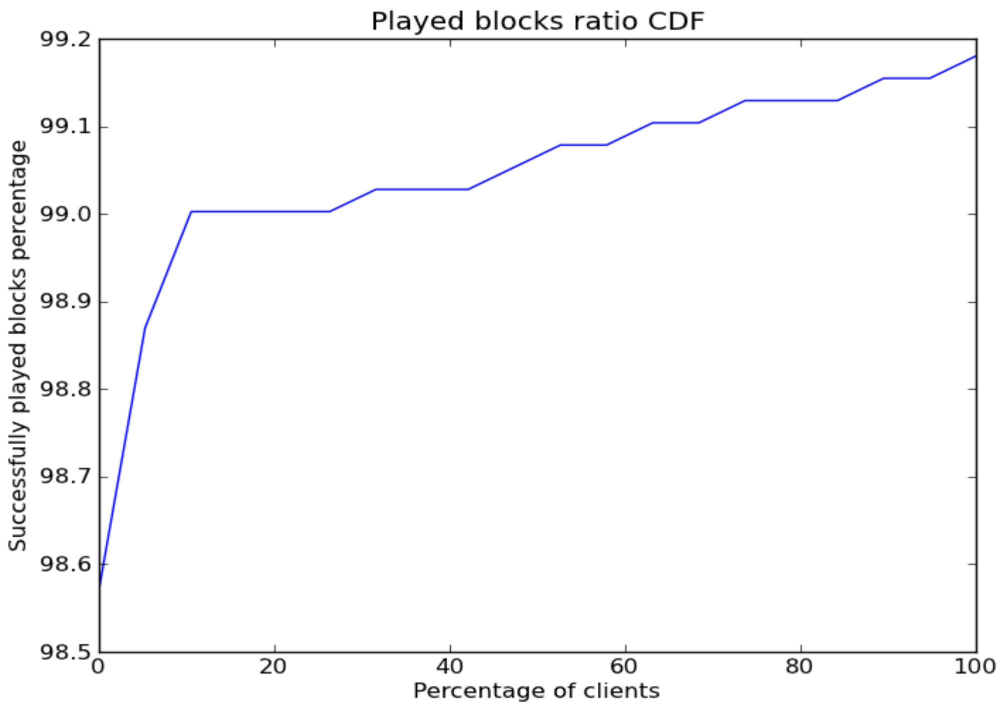

Figure 40: Played blocks ratio CDF

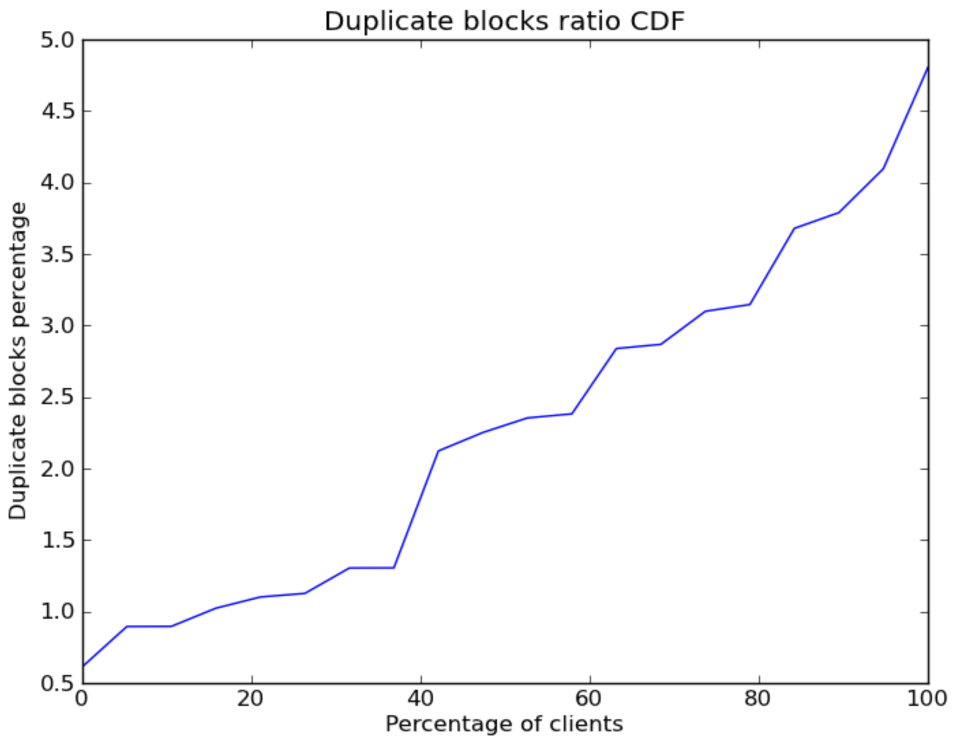

Figure 411: Duplicate blocks ratio CDF

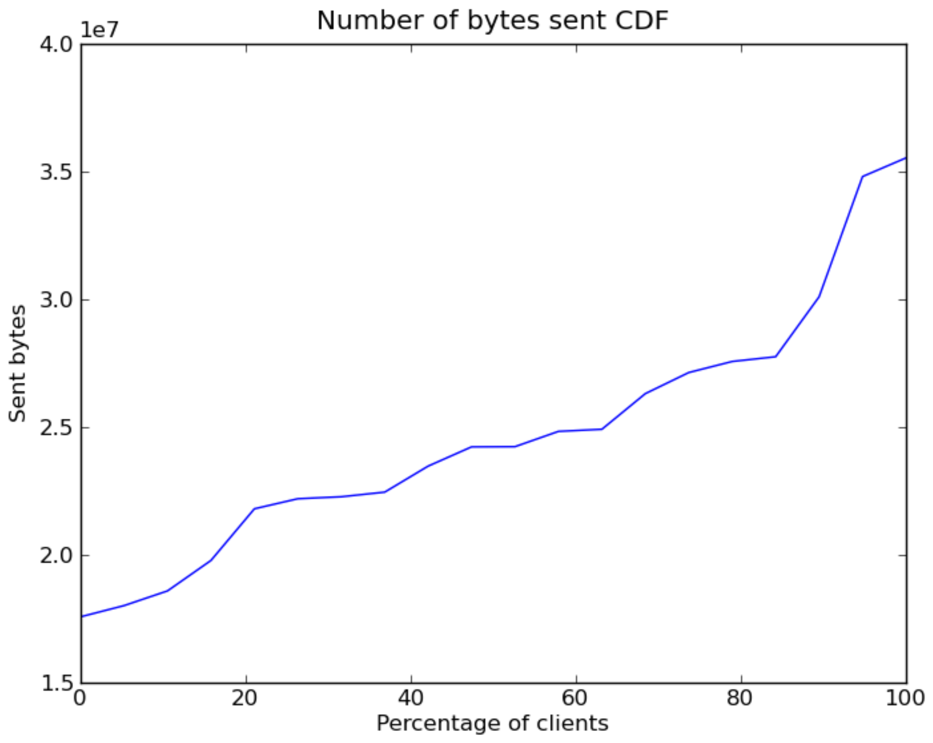

Figure 422: Bytes sent CDF

The execution of this experiment was more close to real user experiment in the sense that the underline network was heterogeneous (Ethernet and wireless), the machines that were used had cpu and memory limitations and the operation of the clients was controlled manually from the GUI of the P2PClient. In order to reproduce the experiment of the 70 VMs, the 20 clients started receiving the video simultaneously.

The figure 40, that depicts the CDF of the played blocks ratio, confirms the high performance of the scheduler according to bandwidth utilization. Specifically, the blocks are successfully played in almost the 100% of the clients.

As far as the idle time and the duplicates are concerned, the behaviour of the clients in this experiment was the same as the experiment of the VMs. Duplicate block percentage was low so the distribution algorithm didn't waste the available bandwidth and the idle time was equally distributed in 80% of the peers.

3.3 Experiment of windows and Linux clients combination

The scope of this experiment was to involve both windows and Linux clients in the overlay and contrast the results with the experiment of only windows clients. The overlay was consisted again of 20 clients, but 15 of them were windows clients and 5 were Linux clients. The experiment was reproduced under the same network conditions and with the same parameters (video rate, upload bandwidth and buffer size). During the procedure of contradicting the collected metrics, we reached the conclusion that the operating system does not affect the data distribution mechanism. As a result, the produced graphs were exactly the same as the graphs presented in paragraph 3.3.

3.4 Real user experiments in UoP campus

The real user experiments took place in the UoP campus and involved a considerable number of students that were invited to download, install and run the UoP P2P client. Although the UoP campus doesn't pose any limitations or restrictions according to available bandwidth, network latency and possible existence of NATs, it can be considered as a reliable testing environment because of its heterogeneous and dynamic characteristics. Specifically, heterogeneity is achieved be the fact that students connected either from the wired network (laboratories, library etc) or from the wireless network offered for free from the UoP campus. Also, the whole environment was completely dynamic regarding frequent arrivals and departures of clients in the overlay.

As far as the experiment, students were invited to participate at a fixed time instance. The transmitted video lasted 1 hour and the bitrate was set to 200kbps in order to contrast the results with the metrics gathered during the laboratory experiments. The P2P client was released for windows platforms and there was an option for students to use either the embedded gstreamer player or the VLC player. During the execution of the experiment clients could arrive and depart randomly according to students' behaviour, so the profile of the overlay was dynamically changed and included 8 to 40 clients.

A logger server was launched for monitoring the whole procedure. The monitoring panel depicted the overlay formation during the experiment as well as many useful real time graphs about successful block reception, duplicates, lost blocks etc.

The outcome of this experiment gave as the opportunity to evaluate the P2P client not only regarding to the data distribution mechanisms but also to the usability of the P2P client itself. The results of the real user experiment are presented in the following graphs:

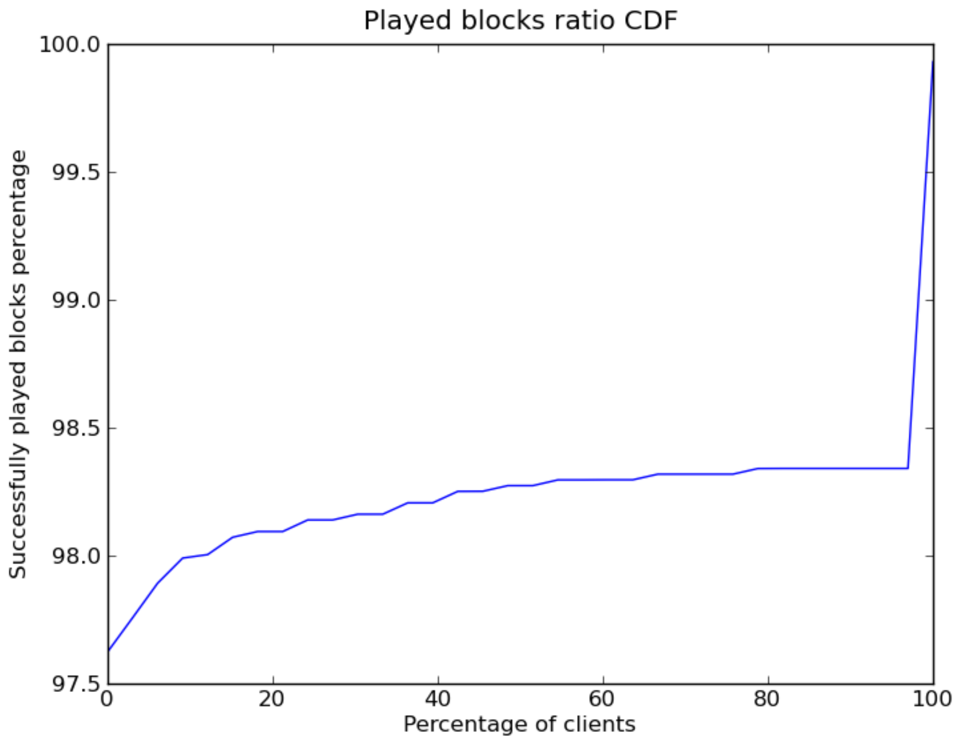

Figure 43: Played blocks ratio (real users)

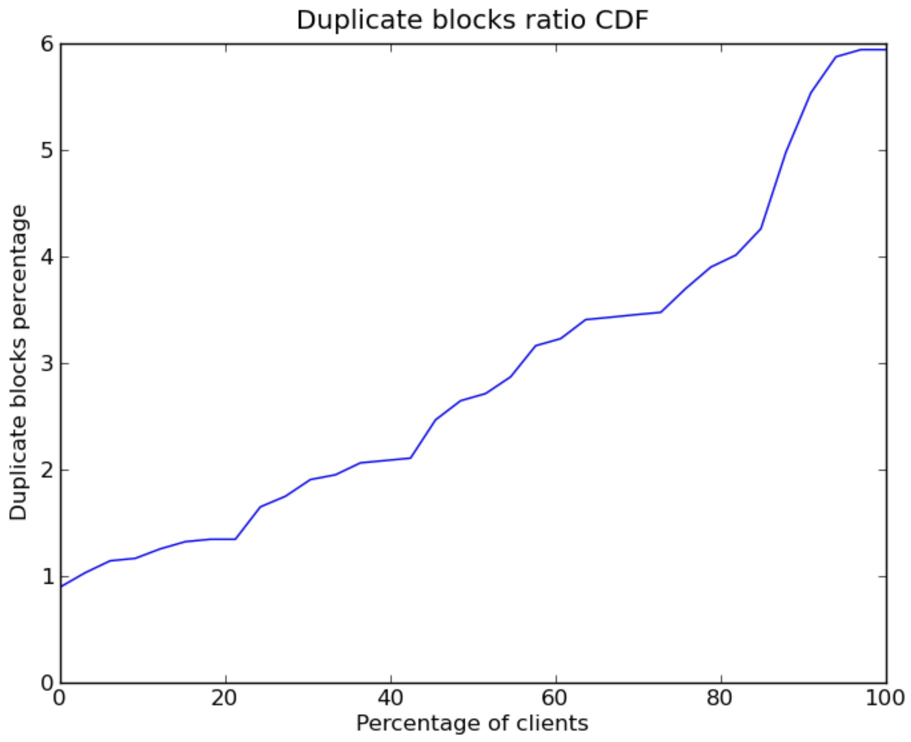

Figure 44: Duplicate blocks (real users)

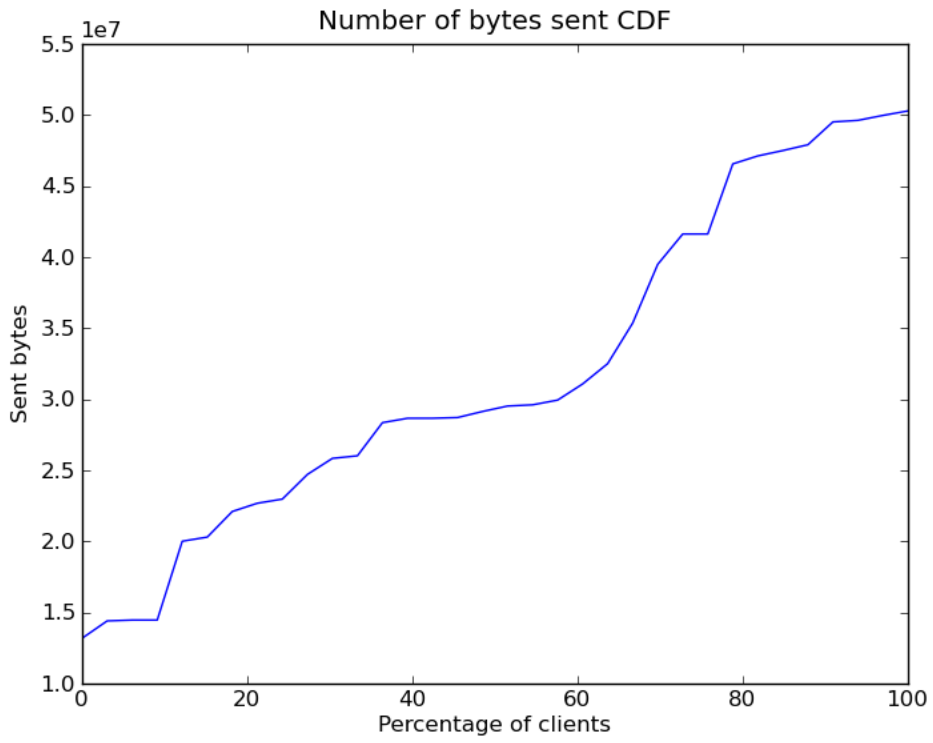

Figure 45: Bytes sent (real users)

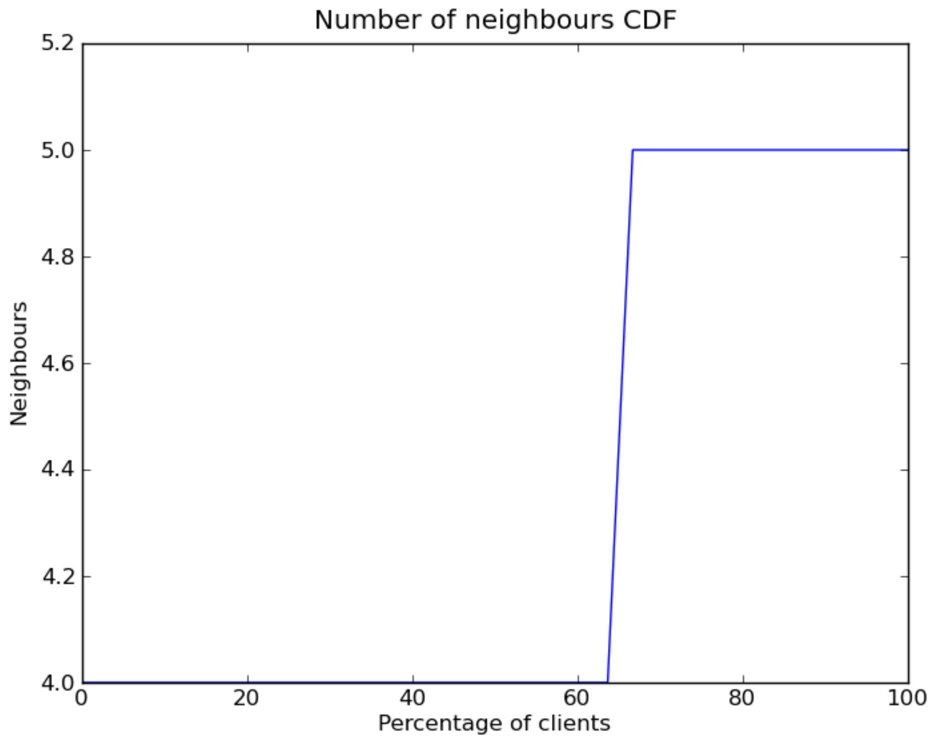

Figure 46: Number of neighbours (real users)

According to figure 43, almost 100% of the clients received successfully around 98.5% of the transmitted blocks, so the performance of the client in terms of bandwidth utilization was high and verified the results in UoP laboratories. This fact was justified by students, which reported the high quality of video playback either in gstreamer or vlc player. Duplicate block ratio was quite low and it verified the excellent response of our architecture to bandwidth waste. Also, the overlay formation was balanced as stated in figure 46 and more specifically every inserted client was joined to 5 neighbours. The bytes sent figure can justify that all the clients contributed to data distribution and the difference in the amount of sent bytes can be explained by the fact that the clients didn’t remain the same time period in the overlay.

4 Conclusions

The overall assessment of the proposed P2P-IMS architecture was performed by evaluating the various sub-architectures separately. This procedure gave the opportunity not only to test the Vital++ components under conditions that verge the real user environment, but also to state the strong and weak points of the whole architecture.

As far as the data distribution mechanism is concerned, it was justified both by laboratory and real user experiments that the quality of service related to the video playback was high and verified the results of simulations. More specifically, users received more than 98% of the transmitted video (total bandwidth utilization) without wasting the available bandwidth (low duplicate block ratio).

In P2PA-SA, the extended sign on procedure caused by the certificate exchange was marginal in terms of time at startup and did not interfere with the other tasks a user endpoint would perform at that stage. Also, the detection of corrupt messages worked well, as all corrupt messages have been detected during tests. The server delay was acceptable as long as not more than about 60 users sign on in the same second, which is quite unlikely, but depends of course on the total number of subscribers and the rate by which they register with the IMS core.

The CS-SA scaled in a predictable manner and the system was performing more efficiently under heavier load. The functionality test caused error rate of 0% and the expected licensing result was returned in all cases where a response was received. The accounting subsystem, which used non-blocking input/output where possible, did not add significant latency to the processing of CS-SA requests. Additionally, the system performed reliably while maintaining connections and state for 50 concurrent clients, despite the exception thrown in 1 out of 400 test cases.

The CI-SA spent from 50 to 100 milliseconds per request. This time period is not the one perceived by the user since there are more delays introduced by the transfer of the message among the IMS components. Although the time required for message processing in the CI-SA is not so long, there may be need for the introduction of parallel processing mechanisms in case the CI-SA is deployed in a setup that involves thousands of users.

The transcoding service result analysis showed that this service can only be used for off-line operations or constraint to services that do not require too many parallel connections and/or manage small file sizes.

The tests that were executed in the DVB-T Vital++ environment justified the operability of Vital++ overlay modules in a real user environment. These experiments carried-out under real transmission/reception conditions and verified the validity of the proposed architecture in terms of reliability and performance.

The BCT Client implementation, integrating also the P2Pengine from UoP, has been evaluated with respect to the system resources that requires and more specifically with respect to memory consumption and its relation with the number of overlays joined for the acquisition of content. The experiment was performed for both vital++ scenarios (Softmix and Live streaming) and the results showed that memory consumption did not exceed the 20MB.

Finally, the OM-SA reacted effectively to client insertions and departures, which fact declares the proper operation of the overlay management algorithm. The algorithm can adjust the overlay formation in a way that does not affect the transmission of the blocks in the running clients. Moreover, the percentage of the successfully played blocks was more than 95% in all the time slots and for more than 90% of the clients.

- End of document -