Enabling live video streaming services realization in telecommunication networks using P2P technology

Nikolaos Efthymiopoulos©(nefthymiop@ece.upatras.gr), Ph. D.; Spyridon L. Tompros© (stombros@keletron.com), Ph.D.; Athanasios Christakidis©, Ph. D; (schristakidis@ece.upatras.gr); Konstantinos Koutsopoulos® (k.koutsopoulos@bluechip.gr), Ph.D.; Spyros Denazis©(sdena@ece.upatras.gr), Professor

©Electrical and Computer Engineering, University of Patras, Patras, Greece
®Blue Chip Technologies S.A., Athens, Greece

Abstract- Peer-to-peer (P2P) systems enjoy wide adoption from internet users. Success of P2P architecture is due to its ability to scale and to organize dynamically the traffic according to the user resources and requests. On the contrary, P2P adoption is constrained in non-real time applications, like content distribution and file sharing. The major reason behind this fact is the inability of P2P to deliver content within user-desired bit rates by making optimize use of network resources. In this paper we focus on the solution and the implementation of technical problems towards a complete system implementation that delivers live multimedia streaming through P2P architecture by the use of IMS. The content indexing and discovery, the scalable management and the distributed optimization of the graph that connects participating peers, the distributed scheduling mechanisms for data exchange, a scalable monitoring architecture and a bandwidth control mechanism constitute the major components towards a complete scalable multimedia streaming system offering streaming stability and high performance.

Keywords: Live Video Streaming, P2P systems, IMS, Content Distribution

I. INTRODUCTION

Deployment of peer-to-peer (P2P) systems in Internet communications faces nowadays steady growth. Because of their efficient mechanisms for flexible content discovery and delivery, P2P systems are in the core of software utilities which today are widely used for file transfer and community-based communications.

P2P live streaming has been a thoroughly researched topic with many contributions from the scientific community, [16][17][20][27][28][29][30][31][32][33] and with many commercial systems [22][23][24][25][26]. Their main objective is the uninterruptable delivery of multimedia streams by means of P2P aiming at high levels of quality of service and experience of the user comparable to conventional content distribution production systems.

The research community has mainly focused on the development of the distributed scheduling algorithms on order create mechanisms that are able to deliver on time every data block to every participating peer. In [30] authors give priority to blocks that are generated recently in order to diffuse them fast to a critical mass of peers. In [32] is presented a distributed algorithm that balances the available bandwidth resources dynamically to the participating peers. In [31]and [16] are developed more sophisticated algorithms for the fast distribution of the stream while it is achieved higher resource utilization. Finally in [17] is analyzed an architecture that prioritize the transmissions of blocks to peers with high upload bandwidth in order to minimize the latency of the distribution of the multimedia stream. In our previous work [21] that is analyzed we present these works in more detail and we also evaluate them. Finally we compare them with a previous version of our distributed scheduling mechanism and we prove that it outperforms.

[16]and [32]propose distributed algorithms that optimize the overlay structure in order to maximize the bandwidth utilization of the participating peers. Our work extends these algorithms and we develop an overlay that is dynamically adapted to the underlying network conditions while it also exploits underlying network topology.

Despite their growing exploitation in non-real- time communications, P2P systems have been very less adopted, if not at all, in the implementation of real-time communications that demand high quality multimedia content, such as video. Main reason of this asymmetric development can be attributed to the inability of P2P systems to cope with the timing requirements that real time communications imply for the network services and the limited perception of content quality and its implications for the quality of the communication link. In simpler words there is an identified gap between variable uploading bandwidth capability contributed by participating peers and the necessary amount of uploading bandwidth that should be contributed in order to meet the bandwidth requirements of the real time multimedia streaming. In this work we introduce a mechanism that monitors the available network resources and calculates the amount of resources that external nodes have to contribute for a stable multimedia streaming.

As new communication services are enabled by the introduction of Next Generation Networks (NGN) technology, the use of the Internet Protocol (IP) becomes widespread in telecommunication platforms and along with it P2P-based applications that are today in wide use in Internet, such as the Skype and Kazaa [1].

A very important issue in telecommunications is the establishment of a "contract" between the user and the network, each time the user is about to communicate using a particular communication method (audio, multimedia, etc). Such a contract guarantees a minimal communication link quality so that the service is perceived by the user satisfactorily. In telecommunication networks such "contract" is called Quality of Service (QoS).

QoS is negotiated between the user and the network at the time of communication link establishment and the method for doing so varies with respect to the access network type (mobile, wireless, fixed). The recently launched first realization of the NGN technology [2], called IP Multimedia Sub-system (IMS), solves satisfactorily the interoperability problems concerning QoS application across mixed type communication networks by introducing generic QoS allocation mechanisms for mobile and wireless, and fixed networks, called Policy Decision Functions (PDF) and Resource Admission Control Subsystem (RACS) respectively [3].

In addition to the uniform application concept of QoS, IMS networks can be deployed on top of mobile, wireless and fixed IP networks without impacting their physical infrastructure, while the IMS services are hosted in the core of the IMS network and therefore can be operated independently from the underlying native network services.

Because of this unique harmonization frame, IMS networks gain steadily in preference among service operators and the first converged communication networks realized with IMS are already launched in Europe, offering IP services that span from simple voice communications to advanced multimedia streaming.

Live video streaming is an important service that several European operators have opted to adopt for the provisioning of Internet TV to their subscribers as an alternative to the cable and satellite TV. As most users prefer watching TV at home, live video streaming is expected to meet demand among users who are behind residential access networks, such as xDSL [4]. On the other hand, live video streaming has stringent timing requirements and high load capacity, which if provided on subscriber basis, must be fulfilled for large stakes of population.

The problem of overloading fixed access network end points at the expense of existing voice and message based communications, has been already obviated in early deployments of Internet TV applications [5]. Is simply the upgrade of access network's communication capacity the only solution to committing QoS? As IP communications diffuse in telecommunication platforms, successful communication paradigms of the Internet world can be re-used to solve inherent telecommunication problems.

In this study we argue that the P2P technology can better serve application of Quality of Service (QoS) in telecommunication networks as an alternative to the fixed end-to-end bandwidth allocation for the entire lifecycle of the service. P2P systems are agile in operation and therefore can be self-organized with regard to the real time traffic conditions in the network so that the content is delivered to user within the predefined QoS settings, thus avoiding permanent reservation of costly network resources.

Our study commences with the description of the frame that allows a P2P overlay network to understand the real time traffic conditions in the paths of the underlying access networks as they are organized in IMS communication platforms (chapter II).

Next, we introduce the client architecture through which live video streaming applications can be offered to users using P2P communication for content discovery, indexing and scheduling (chapter III), while in chapter IV, the mechanisms that sustain live video streaming applications using P2P communication are analyzed in detail. Finally, in chapter V a performance evaluation of our approach is given, showing the ability of the system to meet bandwidth requirements of live streaming applications, followed by some worth-noticing conclusions.

This paper complements our previous work [19], [20] that presents main aspects of our system and the theoretical aspects of our algorithms.

II. ENABLING P2P COMMUNICATIONS REALISATION IN TELECOMMUNICATION NETWORKS

Deployment of P2P systems in IMS networks entails a number of technical challenges, because the decentralized model of peers' communication comes in contradiction to that of the client-server that is currently widely used for telecommunication services realization. Moreover, the charging schemes applied today in telecommunication networks are based on physical link utilization business models that are not applicable on P2P systems, whereas, conventional network resources sharing in the context of telecommunication services, is realized with dedicated QoS reservation mechanisms, not applicable on P2P systems that are highly autonomous and self-organized.

P2P systems are deployed for the realization of user applications that make unstructured use of the network in order to scan, trace down and download user requested content (Figure 1)

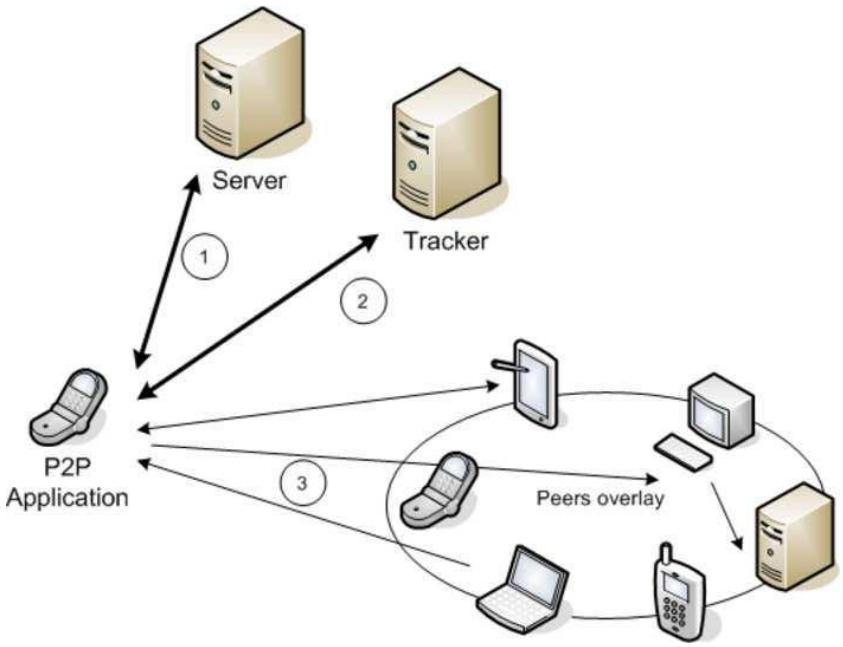

Figure 1: Unstructured communication across P2P systems

As is illustrated in Figure 1 P2P applications and supplementary components interact in order to discover the source of the requested content. Firstly, the P2P application running on the user terminal communicates with a *P2P server* in order to get content description, which usually consists of a file name, size, a hash table and the address of the *tracker* (flow 1). The tracker is a special server that contains descriptions of peers, which currently perform downloading of the same content or already have pieces of it (flow 2). Once peer information concerning the content in question is retrieved from the tracker, the P2P application of the user terminal starts communicating with the list of peers in order to get the content fragments and use them to reconstruct the content (flow 3). An important issue with regard to this communication principle pertains to the peer selection process that the user application performs in order to choose out of the available peers those having the best characteristics concerning content demand. As we show in chapter IV, these characteristics vary with respect to the type of the user application. For example, when implementing simple FTP applications the only choice criterion is the gathering of all content fragments, while in the case of live video streaming, the selection criterion also includes metrics about intermediate link performance and peer downloading capability.

As P2P systems are inherently IP-based communication overlays, mapping of their communication logic on IMS network infrastructures can be realized utilizing dedicated Application Servers Figure 2. In IMS networks, Application Servers (AS) are used for hosting services call control logic. Each service requires call control procedures realized on a dedicated server, while user applications wishing to use a particular service send their signaling messages to the dedicated AS that implements the given service. Even when P2P services are about to be utilized, users must previously be recognized and authenticated by the network for service access, using standard IMS signaling. Then, the user is allowed to make use of a P2P service by contacting the P2P-AS that stands between the IMS network and the P2P overlays as signaling messages router Figure 2.

To enable hosting of various P2P-based services, the content indexing capability of the P2P-AS mentioned earlier in the reference communication scenario is hosted on separate Content Indexing-ASs, each one of them providing a list of available trackers dedicated to servicing the given application. This way, an IMS network is capable of hosting any P2P-based service as long as an appropriate Content Indexing-AS exists.

The P2P-AS maintains communication between the user, the Content Indexing-AS and the available trackers (tracker-AS). In addition to this function, the P2P-AS provides the means for implementing telecommunication oriented functions concerning P2P service usage, such as charging and content access rights, both of which are implemented with dedicated servers (charging- and DRM-AS).

Communication between the users and the application servers implementing components of the P2P overlay system is realized using the Hypertext Text Protocol (HTTP) [6] and the Session Initiation Protocols [7]. Before content delivery takes place, P2P application users should first get connected to the P2P-AS, get routed to the right Content Index-AS, where the list of trackers available for the given application type is maintained and finally the most close to the user tracker is called to hand over the peer list.

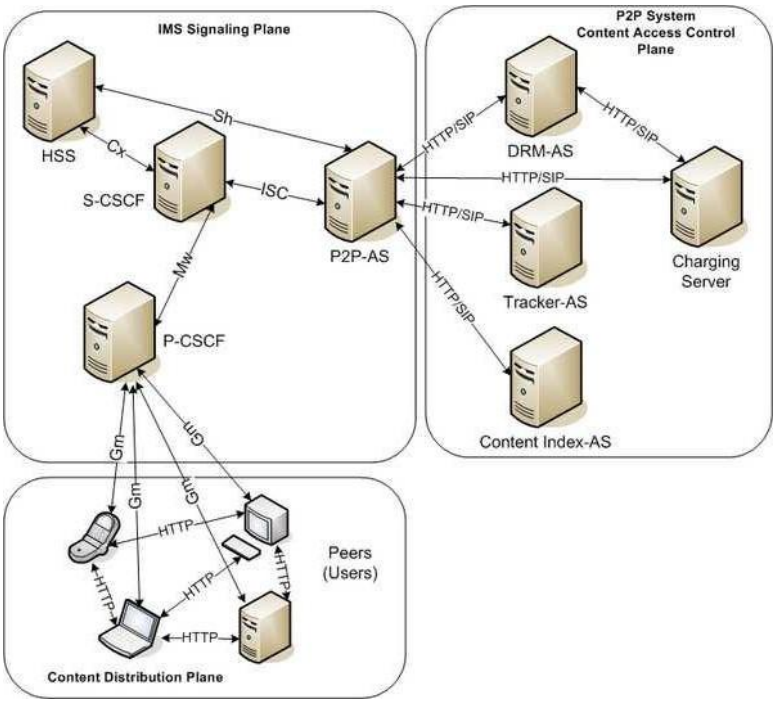

Figure 2: Architecture for integrating P2P applications in IMS networks

Once the content list is obtained, the user application establishes connections with the target peers and downloads the content. Figure 3 illustrates an example of such operation. For simplicity reasons the messages related to IMS network operations are suppressed.

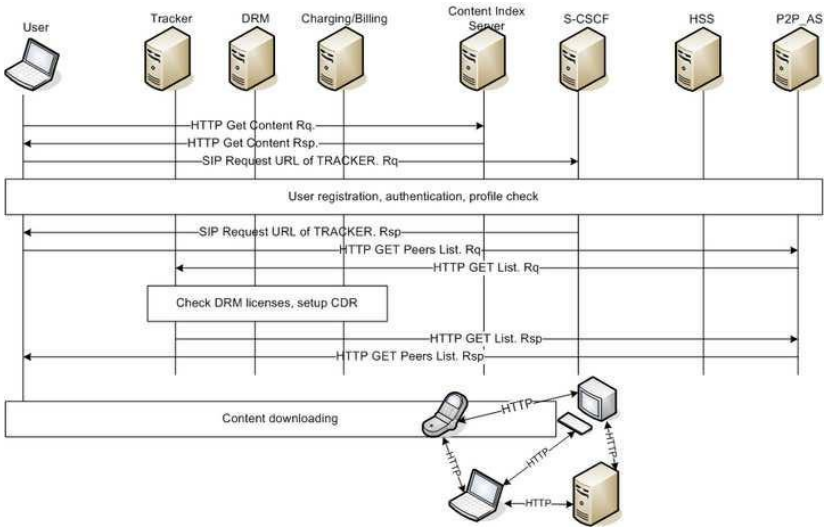

Figure 3: Message flow of P2P systems integration in IMS networks

Initially the user contacts the Content Indexing-AS to get the file that describes the content and the address (URL) of the tracker server (HTTP Get Content Rq/Rsp.). Using the returned tracker address, the user asks the IMS network to provide the list of peers having parts of the requested content (SIP_URL_TRACKER_Rq.). The latter request triggers the process of user registration and authentication in the network to take place, whereby the user is authorized for accessing the requested P2P service. This operation involves the user, the S-CSCF and the P2P-AS. After the authorization process is successfully accomplished (SIP_URL_TRACKER_Rsp.), the user is allowed to communicate with the tracker (HTTP GET Peers List Rq/Rsp.) and the rest P2P servers of the network (HTTP GET List. Rq./Rsp.).

Upon receiving the request for the peer list file download, the tracker asserts the DRM settings of the requested content and presets accordingly the Charging Data Records (CDR) of the given user contained in the charging/billing server. Finally, by getting the peer list (HTTP GET Peers List Rsp.), the user is able to communicate with the peers in order to download the requested content.

III. P2P CLIENT ARCHITECTURE

Realization of P2P communications in telecommunication networks demands that user terminals and network nodes can communicate with each other using both centralized and decentralized means of communication.

Setting in this context as main focus the study of live video streaming services, we are able to identify a number of mandatory signaling procedures for the parallel management of communication within the frame of the IMS networks and across P2P overlays:

- User authentication and authorization.
- Content publication and discovery.
- Session establishment and maintenance.
- Content streaming.

For each of these functionalities a solution can be provided with mechanisms defined per communication plane (Table 1).

	SIP/IMS Plane	P2P Plane
Authentication/Authorization	Registration	DRM, password based encryption of content, peer to peer authentication
Content Publication/ Discovery	SIP Presence, Custom SIP Application Server based on SIP Instant Messaging	Distributed Hash Table Network -DHT, Server based discovery of peers, Offline torrents
Session Establishment/Maintenance	SIP INVITE, Custom SIP Application Server based on SIP Instant Messaging	Gossip Protocol
Content Transfer	RTP	P2P links

Table 1: IMS/P2P mandatory functionalities and support mechanisms

As discussed in the previous chapter, our P2P client architecture depicted in Figure 4 is influenced by the concept that the server/client-based call control functionality related to service selection and QoS establishment should be handled by the IMS network procedures using the SIP protocol, while content management can be realized using P2P means of communication.

Access to P2P services from an IMS client is enabled by retaining the IMS procedures of user registration in the network and content publishing/discovery and replacing those of content streaming with decentralized ones.

Application of Quality of Service (QoS) is implemented separately for the two communication planes. On the IMS communication plane, the SIP protocol is used for the establishment of QoS in end-to-end manner, while on the P2P communication plane the overlay uses a gossip protocol [8] that allows it to calculate whether the QoS defined at the IMS plane is satisfied given that the content can be downloaded from a particular list of peers.

To cope with the diverse requirements of the two contradictory communication models, the client architecture follows a modular design consisting of three layers.

The top layer (*User Interaction Layer*) is the same for all user applications and provides APIs to the call control and media protocols of the layer 2 (Services Provisioning and Protocol Integration Layer). The control part of the user interface provides user applications with functions for content searching and publication, while the media management part provides functions for media presentation and capturing. The layer is independent form the underlying service provisioning and transport layers and hosts both media management and call control functionalities.

The media management functions implement interfaces to the mechanisms required for media capturing, reconstruction, and playback using the available JMF libraries, whereas the control functions implement interfaces to the logic of the services provisioning layer for user registration, content management (discovery and publication) and communication with the P2P overlays concerning their selection and real time management during content delivery.

From implementation point of view, both functionalities have been developed as plug-ins for web browsers so that they are operating system (OS) independent and can be exploited transparently to the user terminal.

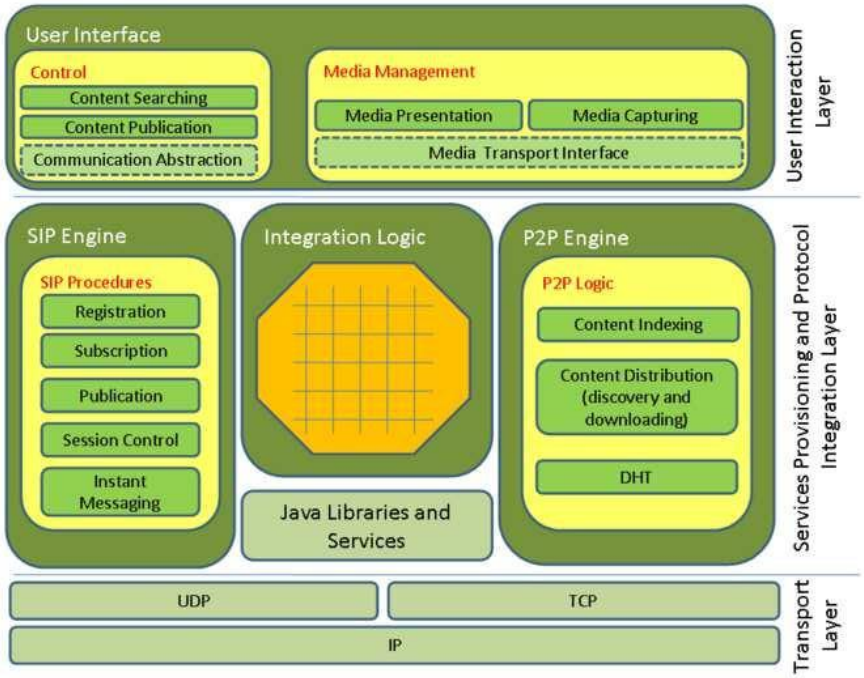

Figure 4: IMS/P2P client architecture

The layer 2 (*Services Provisioning and Protocol Integration Layer*) of the client is divided in three planes. The first plane, called *SIP Engine* hosts the call control protocols for IMS operations implementation, the third plane, called *P2P Engine* hosts the signaling protocols for P2P communications implementation, whereas the intermediate *Integration logic* plane hosts the interfaces needed for the communication of parameters between the SIP and P2P Engines. The SIP engine implements IMS network logic for user registration, and session and content management, while the P2P engine realizes overlay, transport management and DHT [9], concerning the user required content.

Moreover, the P2P engine utilizes a number of Java-based interfaces that allow peers to be selected and managed, using the traffic parameters established by the SIP engine, such as the QoS settings and the content type. In addition, communication between the peers composing the peer overlay and the client can be implemented using SIP Instant Messages (IM). In that case, peer information is passed to the P2P engine via the same interfaces of the integration logic plane.

The integration logic plane implements Java-based interfaces for the communication between the SIP and P2P engines. This plane is functionally independent from the other two planes and therefore can be on-the-fly modified if additional communication logic between the two engines is needed.

The layer 1 (*Transport Layer*) consists of transport protocols that are used for signaling and data traffic transportation. This layer is the container of the transport protocols needed for both the IMS and P2P communication planes. Although the protocols contained in this layer are predominantly of IP type, narrowing down selection choice of layer protocols significantly, the client architecture is generally open to any higher layer protocol, with indicative examples being the UPnP [10] that allows device capabilities identification and the realization of the gossip protocol that peer overlays use in order to maintain internal integrity. The protocols of the transport layer are integrated with the upper layers using standard Java APIs [11].

Overall, our client implementation is Java-based, with the media handling of transport layer relying on the use of the Java Media Framework (JMF) and the higher layer SIP, P2P engines and integration logic planes being implemented as Java libraries so that they are interchangeable.

IV. LIVE VIDEO STREAMING SERVICE REALISATION

Video streaming applications require functions for content indexing and distribution. Although these functions are needed for every P2P service realisation, here both of them must be accommodated with particular timing requirements that emanate from the QoS settings of the user.

The rest of this section is divided in four parts that represent the major functionalities of our system. The first is the content index sub architecture that allows the insertion and the discovery of multimedia objects in our system. The second is the overlay management functionality that enables the dynamic optimization of the overlay. In this part we present in detail information relevant with the distributed implementation of our algorithms. The third is our P2P block scheduling mechanism for fast and reliable delivery of each block to each participating peer. The fourth part presents our monitoring scalable algorithm that allows the control of the available network resources.

A. *Content Indexing*

Content indexing in P2P overlays operating over the Internet is performed using *Presence Servers*. Today exploitation of presence servers is widespread between P2P applications and their integration with user applications using the SIP protocol is well described in IETF RFCs [12].

By subscribing to Presence Servers, peers can be informed about events that help them to communicate with each other concerning content management. On the terminal side, the user can register with a presence server in order to be able to receive notification regarding the availability of particular content.

Since in P2P systems content is not downloaded from a single source as a whole but in fragments from multiple sources, organisation of content fragments availability is made using the concept of *super peers* [13], which point out peer communities that contain fragments of the same content. A super peer is actually another presence server dedicated for serving particular content, for example a movie. To such super peers subscribe peers that contain fragments of that particular content. In a P2P system the role of peer has the end user, a server in the network or any other form of content source (Figure 5).

As super peers can be formed dynamically depending on the timely availability of content sources in the network, they are accessible by users through the presence servers of the network. For example, a presence server may be grouping super peers of certain content type, i.e. action movies. Super peers are making themselves available by registering their content with the presence server. In the same fashion sources of content register with the super peers. Both publication and publication removal are performed using the SIP PUBLISH method, while content request by users looking for particular content can be denoted using the SIP SUBSCRIBE method.

Following this hierarchical model involving presence servers that register dynamically variable super peers, a video streaming client can get a real-time picture of where and which content fragments are available on peers in the network by getting subscribed to the nearest presence server. In an IMS network the presence and the super peer servers are the equivalents of the content indexing server and the tracker.

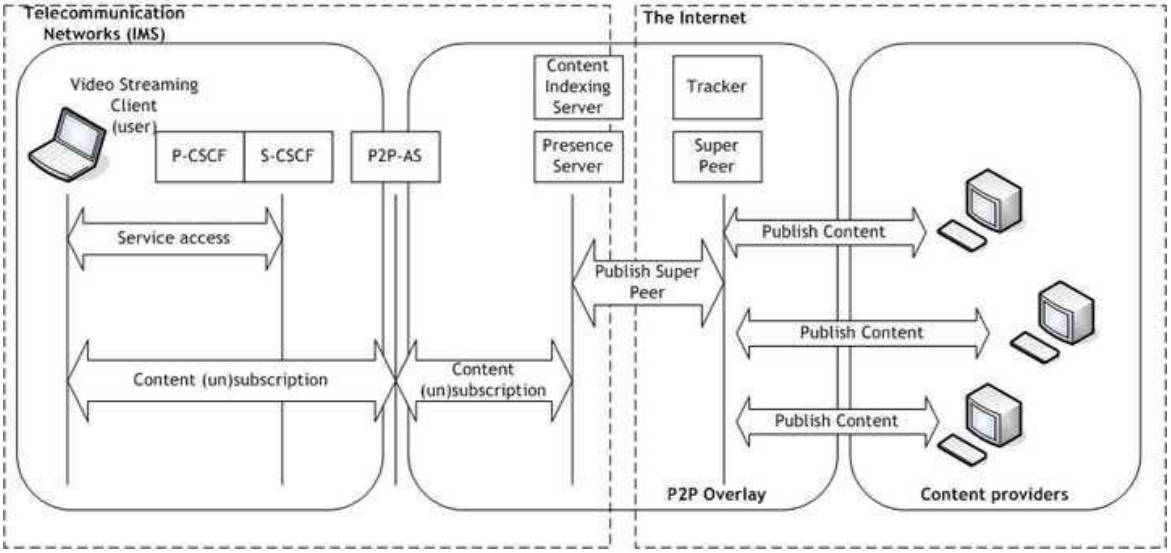

Figure 5: Reference communication scenario of achieving access on distributed content from an IMS network

B. *Overlay management*

An overlay graph architecture that forms the substrate for an efficient P2P live streaming system should meet the following requirements.

Firstly, the overlay graph should be constructed in such a way that every peer has a sufficient number of neighbors proportional to its uploading bandwidth. This guarantees optimal utilization of each one's uploading capability which, in turn, has a positive impact on block scheduling. Likewise, each node should have a sufficient number of incoming connections for the undisruptive reception of the video stream regardless of the dynamic network conditions and/or peer arrivals and departures. In addition, the overlay should be dynamically reconfigurable in order to dynamically react to the various changes of the underlying network as well as the dynamic peer behavior. Last but not least, it should exploit the underling network latencies, i.e. round trip times, between peers, meaning that each peer should have as its neighbors those peers that are close to him in the network. In other words, the overlay must reflect as much as possible locality information in the way that peers are kept organized. Our proposed overlay architecture derived from the aforementioned requirements is depicted below (Figure 6).

We distinguish between two types of peers: the super peers and the slow peers. The former are those peers with uploading bandwidth higher than the service rate of the video server (video playback rate) whereas the latter are peers with upload bandwidth less than the service rate.

Every slow peer that joins the system becomes part of a *base overlay* and is assigned a fixed number of neighbors, say M_B. This is a bidirectional mesh overlay, balanced with respect to the number of neighbors. Otherwise, if this peer happens to be a super peer then it becomes part of a different overlay, called *super-peer overlay*, of similar characteristics as the base overlay. In this overlay, the peer is also assigned a fixed number of neighbors, say M_S.

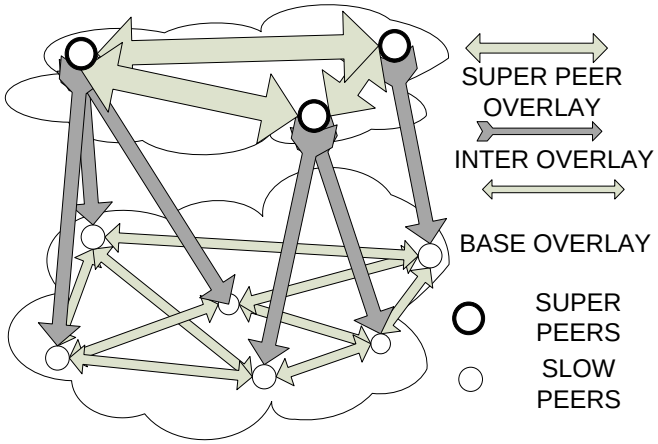

FIGURE 6. The structure of the overlay graph.

A third overlay, called *interconnection overlay*, connects the base and super-peer overlays by assigning a number of super peers to each slow peer. More specifically, each slow peer in the base overlay selects a fixed number of super peers, M_I, which wishes to connect with. These interconnections are unidirectional originating from the super peers (outgoing reconnections) and arriving at the slow peers (incoming connections). They are also distributed among super peers in a manner proportional to the excess of their uploading bandwidth (uploading bandwidth minus the stream service rate). The quantities M_B, M_S and M_I are parameters of the system overlay. Figure 6 depicts such a system overlay with $M_B=3$, $M_S=2$, and $M_I=1$.

a)Distributed Optimization Algorithms (DOAs)

The proposed system overlay (base, super-peer and inter overlays) is continuously organized according to two distributed algorithms: a) a locality aware intra-overlay distributed optimization algorithm (Intra-DOA) applied to the base and super-peer overlay, responsible for organizing peers with M_B or M_S neighbors, respectively, and b) an inter-overlay distributed optimization algorithm (Inter-DOA) algorithm responsible for the interconnection of the base and super-peer overlay with M_I connections per every slow peer while taking into account the network latencies (locality) between the peers.
These algorithms aim at rendering the overlay dynamically manageable with respect to the path latencies in the underlying network and to the dynamic changes in the peer population. To achieve establishment of low latency path between peers these algorithms create and maintain overlays where nodes have the appropriate number of neighbors, whereas each node is as close as possible to its neighbors topologically.

The DOAs makes use of a function, called *energy function*, denoted as E(i,N(i)). It expresses the energy of node i in relation to the set of its neighbors, N(i). The energy function is defined as the sum of network latencies, Stt(i,j), of node i with all of its neighbors $j \in N(i)$, that is,

$$E\big(i,N(i)\big) = \sum_{j\in N(i)}^{|N(i)|} Stt(i,j) \quad (1)$$

Furthermore we define E_{all} as the total energy of the nodes that participate in the overlay. Hence,

$$E_{all} = \sum_{i\in S}^{|S|} E(i,N(i)) \quad (2)$$

The set S includes all the nodes that participate in the overlay every time instant. Our algorithms are iterative distributed algorithms that each node executes periodically and use integer linear optimization to minimize the sum of the energies of a small fraction of nodes that participate in an iteration, while simultaneously setting the number of neighbors of the participating peers to their desired value. Detailed rational about the architecture of our overlay as well as the properties of the DOAs can be found in [20].
An example of the intra-overlay distributed optimization algorithm is shown in Figure 7. Peers NODE1 and NODE2 execute the algorithm in order to reorganize their neighborhood. In the figure the length of the edges between two nodes is proportional to the

network latency between them. The left part shows the state of the neighborhood before the execution and the right and state after the execution of the algorithm.

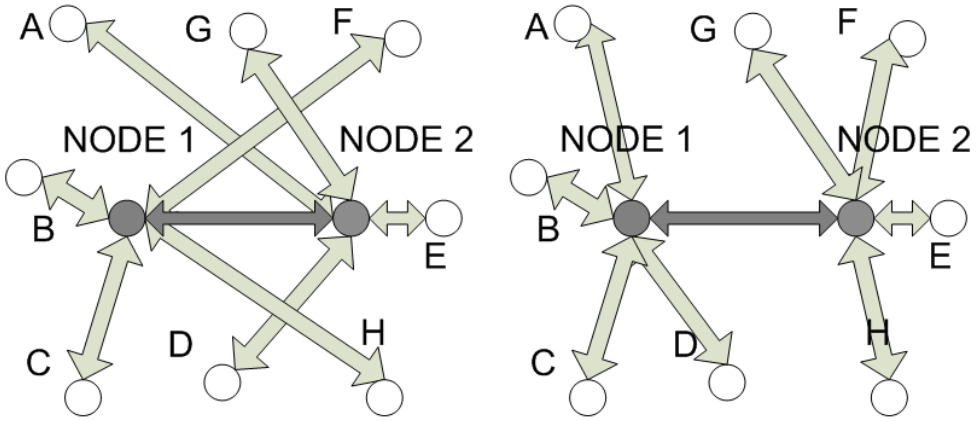

Figure 7 Overlay organization before and after an execution of Intra-DOA

An example of the inter-overlay distributed optimization algorithm is shown in Figure 8. Again peers NODE1 and NODE2 execute the algorithm in order to reorganize their neighborhood. In this example peer NODE1 has twice the uploading bandwidth of peer NODE2 and that is why in the right part of the figure, which depicts the state of the neighborhood after the end of the execution, peer NODE1 has double neighbors compared to peer NODE2.

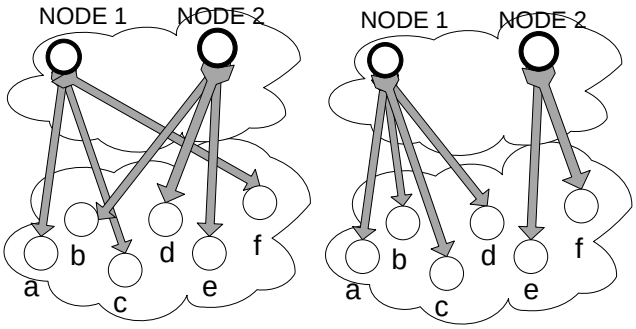

FIGURE 8. Inter overlay connections before and after Inter-DOA execution.

b) Implementation of DOA

In this section we will describe the implementation of the distributed algorithms needed for the realization of the proposed overlay architecture.

As it was described above, our overlay architecture consists of two bidirectional overlays, the base and super overlays, in which every node has the same number of connections, and of an unidirectional overlay, the interconnection overlay, in which every slow peer has the same number of incoming connections which are distributed to the super peers in a manner proportional to their uploading capabilities. Moreover, in order to have overlays with neighborhoods of minimum energy, the two overlays distributed optimizations algorithms have to be implemented.

We split the procedure of building our overlay architecture in two processes. The first one is the creation of the overlays, which consist of the insertion and departure algorithms, and the second is the reconfiguration of the overlays, which consists of the two DOAs.

Whenever a slow peer enters the base overlay it randomly selects $M_B/2$ peers from the base overlay as its neighbors. By doing this, a total of M_B new connections are created leaving the sum of connections in the base overlay constant. Similarly when a super peer enters the super-peer overlay takes as neighbors $M_S/2$ peers. Finally, a slow peer takes as incoming neighbors M_I peers (interconnections) from the super-peer overlay.

On the other hand, when a peer leaves, its neighbors replace this peer with another one with probability fifty percent, except when a slow peer loses its interconnection with a super peer in which case it replaces the departed node with another one.

The reasoning behind this mechanism is to keep the number of overlay connections constant and rely on the DOAs to distribute them to the participating peer in a manner that satisfies the given specifications.
In this point we will describe the implementation of the two distributed optimization algorithms. As the principle of the inter and intra algorithms are quite the same we will focus on the implementation of the intra overlay distributed optimization algorithm which is executed by the peers of a bidirectional overlay in which every peer should have exactly M neighbors. We remind that the goal of the algorithm is the creation of neighborhoods with the minimum energy, as it is defined in the previous section, and the reorganization of the overlay in such a way that every node has the same number of neighbors.
An example of the execution of the algorithm is shown in Figure 7. In this example NODE 1 begins the execution of the algorithm and is defined as the first active node. NODE 2 is the peer which NODE 1 chose to execute the algorithm with and is defined as the second active node. The neighbors' of those two peers are defined as satellites.
The implementation of the algorithm has to be done in such a way that

- The fast convergence of the overlay to the optimum state is ensured
- The consistency of the graph is guaranteed
- The required bandwidth for the execution of the algorithm is minimized

The fast convergence of the overlay to the optimum state assumes the parallel execution of the algorithm by the participating peers. As there is none central entity to synchronize the parallel executions, each peer must periodically try to execute the algorithm, with the duration of that period being as small as possible.
However, the successful execution of the algorithm by a peer results in the redistribution of the peer's neighborhood connections. As a consequence, there is the possibility of damaging the consistency of the graph in cases where two adjacent peers execute the algorithm in parallel, because the peers of that neighborhood might have conflicting perspectives about the state of the neighborhood's connections after the end of the executions.

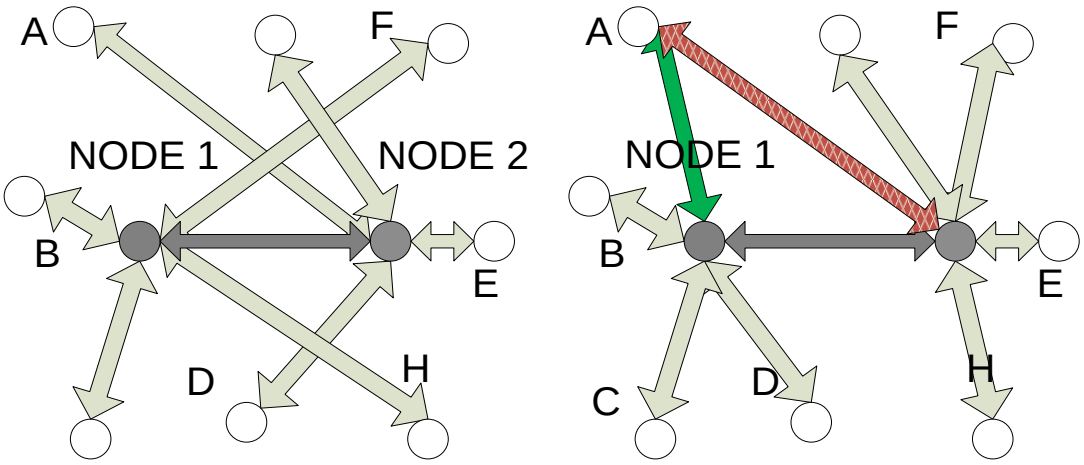

Figure 9. Counter example of Intra-DOA execution

Such case is depicted in Figure 9 where peers NODE 1 and A execute the algorithm simultaneously. In this example peer NODE 1 decides that he should take as neighbor peer A and so he makes a new connection between them while removing the existing connection between peer A and peer NODE 2. However, peer A decides that he must keep the connection he already has with peer NODE 2, which in that time peer NODE 1 decided to remove. The result, which is depicted in the right part of the figure, is that peers NODE 1,NODE 2 and A have conflicting perspectives about the state of the neighborhood which means that the consistency of the graph has been compromised.
For the avoidance of such problems, the distributed implementation of the algorithm has to guarantee that only one node is responsible for the state of a connection in every instant. Combining this requirement with the fact that only active nodes can alter the state of a connection we conclude to the following rules:

- A peer can become active only if he is not a satellite in a parallel execution of the algorithm by another peer.
- A peer can be satellite in multiple parallel executions of the algorithm by other peers.

These two rules guarantee that the parallel executions of the algorithm by different peers, which is required for the fast convergence of the overlay, doesn't affect the consistency of the overlay. In order for a peer to keep track of the number of parallel executions of the algorithm in which he participates as a satellite, he maintains a counter, defined as lock counter. A node can become active only when the value of that counter is zero.

Moreover, in order for a peer to execute the algorithm he must check if his neighbors can take part in that execution as satellites. As the execution period of the algorithm is low, in order to have a fast convergence of the overlay, the possibility of

some of his neighbors being active is great. For this reason we don't set as a requirement the participation of the whole set of a peer's neighbors in the execution of the algorithm from that peer as satellites. On the contrary, a peer can execute the algorithm reorganizing only the connections between him and those neighbors that aren't themselves active as well.

Having explained the principles for the implementation of the distributed optimization algorithm we continue by describing briefly the steps of the execution of the algorithm by one peer.

- When the timer of a peer expires then he becomes the first active peer if his lock counter is zero and his is not already active. Otherwise he resets his timer.
- The first active peer chooses a random neighbor to act as the second active peer. If that peer is not active in another execution he sends back to the first active peer his routing table, otherwise he rejects the invitation and the first active peer become inactive again and resets his timer.
- The first active peer sends to the second his routing table.
- The two active peers contact each other's neighbors in order to inform them of the imminent execution of the algorithm, and also measure their latencies with them in the process.
- The neighbors of the active peers respond negatively if they are active themselves, otherwise they become satellites and increase their lock counter.
- When the second active peer has gathered all the answers from the neighbors of the first active peer, he sends to him a table with the latencies between him and those peers, as well as the information about their involvement in the execution process.
- When the first active peer has received the answers from the neighbors of the second active peer and the updated table from the second active peer he has all the information needed in order to execute the optimization algorithm. The output of the execution is the final routing tables of the two active peers which he also sends to the second active peers.
- The two active peers inform their neighbors about the output of the execution and set the state of their connection accordingly. Those neighbors decrease their lock counter if they had acted as satellites.

C. *P2P block scheduler*

In a P2P live streaming system a source generates a video/audio stream with a service rate of μ *bits/sec*. Assuming that every second of the stream is divided into N_b blocks, each block is generated every $1/N_b$ *seconds* with a size equal to $L_b=\mu/N_b$ *bits*. These blocks are diffused to a small subset of peers. Consequently, peers that are neighbours in the overlay exchange blocks mutually until they acquire the whole stream. This exchange is carried out according to a block scheduler that runs in every peer and maintains a buffer of all N_b*t_s blocks generated within a sliding window of t_s seconds (with t_S we denote the setup time). Two states are of interest: *received blocks* and *missing blocks* (blocks that have not been delivered yet). Periodically each peer propagates its buffer to all of its neighbors and the scheduler that runs in every peer decides which block should be transmitted next and to which neighboring peer.

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

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

Taking into account the objectives above, our scheduler is composed of three components. The first is the *token generation algorithm* where each potential sender periodically defines the set of the potential receivers that is able to serve and issues tokens to them. The second is the *proactive block request algorithm* where a potential receiver matches the tokens that it has received with different blocks that it misses. We call the period in which these two algorithms are executed, `request_interval`. The third component is the *neighbour selection algorithm*, which is executed by the sending peer just before transmission of a block

takes place, taking into account the requests from receiving peers, their upload bandwidth capabilities and the amount of blocks they are missing.

a) Token Generation Algorithm

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

$$|\text{token\_set(i)}| = c(i) * \text{request\_interval}/L_b \quad (3)$$

A key requirement in order to have a P2P live streaming system with high bandwidth utilization and timely stream distribution is to uniformly distribute the sum of these tokens to every participating peer since everyone of them has to acquire blocks with a rate equal to N_b. We denote the probability in which a sending peer `i` selects a peer `j` in the `token_set(i)` as `P(i,j)`. Initially, the probabilities for each peer `i` are assigned as `P(i,j)= 1/M(i)` for each `j`.

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

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

b) Proactive Block Assignment and Request

This component offers a function that proactively determines which block a peer should request from its neighbors that have already sent a token to it during the last `request_interval`.

The execution of this algorithm takes place periodically within the same period as the token generation algorithm. Therefore, the block request from each potential sender is piggybacked to the same message with the token transmissions, thus avoiding unnecessary bandwidth overhead.

The block assignment process is accomplished with the execution of a matching algorithm between the missing blocks of the peer and the neighbor peers that have already requested the same blocks. This practice reduces effectively duplicate block transmissions as the newly produced and rare blocks have a high probability to be requested by other neighbor peers and therefore can be provided to the target peer faster by the neighbor peer.

c) Neighbour Selection for block transmission

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

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

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

The node selected for block transmission is the one with the maximum *d(i,j)* $\forall j$ *that has issued a request to node i.*

In this equation, the difference of i,j expresses the number of blocks that the sender currently possesses and the receiver misses. `|neighbors(i)|` denotes the total number of neighbours of node *i that have made a request to* `i`. `Rank(i,j)` is a function that returns the position of node *j* in a list with neighbours list in descending order with respect to their uploading capability. We have chosen to model the network bandwidths in this way in order to make our scheduler independent of the uploading bandwidth data set and as such suitable for every uploading bandwidth distribution.

The `buf_size` is equal to $N_b * t_s$ and denotes the number of blocks that nodes exchange at each time instant. Finally, the parameter `per` is a constant value representing the percentage of the buffer size. As is explained below, we have successfully experimented with values of parameter `per` that are between 5%-10% of the buffer size.

If we examine the second term of the decision function, we note that it is a linear function of rank(i,j) because assuming values in the range of [0,1], we have 0<rank(i.j)<=|neighbours(i)|. When nodes have small differences concerning missing blocks, the first

term of the equation is very small and so rank(i,j) has a dominating effect on the selection of node j and so the diffusion of blocks is done by nodes with high uploading capability. On the other hand, when differences for missing blocks in the order of per*buf_size are observed, our scheduler approximates the most deprived scheduler behaviour with difference (i,j) becoming the dominant parameter for selecting node j. In this way we guarantee high degree of fairness in blocks distribution.

D. *Management of QoS in P2P systems*

A P2P system becomes capable of maintaining QoS using a manager that ensures that for each distributed file or video there are enough network resources for its accomplishment. Sometimes network resources are adequate, while in other cases it turns out that resources are not sufficient. For example, an overlay where nodes have an average upload bandwidth of 900 kbps is not able to deliver to its peers a video stream with 1000 kbps.

To cope with the latter case we have developed two mechanisms that will further enforce the capabilities of our system towards QoS optimization. The first mechanism pertains to additional provision of resources taken from centralized servers. Choosing either of these two mechanisms, a P2P system can accommodate diverse QoS requirements, without network nodes modifications. The QoS mechanism is responsible for dynamically provisioning the correct amount of bandwidth resources for the uninterrupted distribution of each object. It is comprised of two components: the monitoring component and the resource allocation component.

The **monitoring component** is responsible for monitoring the resources of each overlay. In more detail these are the sum of the upload bandwidth that has the set of peers that participate in the distribution of each object. This component also calculates dynamically the difference between the available upload bandwidth and the required upload bandwidth for the distribution of each object. So a server samples periodically a small subset of the participating peers from each overlay and estimates the mean of their incoming flows in bytes and the mean of the time in which they were transmitting content during that period. The attributes of P2P block scheduling algorithm are useful for a very good approximation of the average value of upload bandwidth that sets of peers in each overlay have. Using these values, the monitoring component is able to calculate the average upload bandwidth of the peers and thus the minimum additional bandwidth, if needed.

The **resource allocation component** uses a set of bandwidth provisioning servers, which can control the upload bandwidth. These have been designed to comply with ETSI's Resource and Admission Control System (RACS) functionality, first defined in the 2006 release of their NGN specifications. Having as input the output of the monitoring component, the QOS resource allocation component provisions the excess bandwidth that is required in such a way that each server is connected with peers, which belong to the same ISP (if possible) in order to minimize the inter-ISP traffic.

V. PERFORMANCE EVALUATION

For the evaluation of our P2P streaming application system we have used the OPNET Modeler v.14 [14], with which we evaluated the performance of our algorithms described in previous section under various underlying network topologies and conditions. In order to model the heterogeneous uploading bandwidth capabilities that usually peers have, we set uploading bandwidths equal to 4000 kbps (class 4), 1000 kbps (class 3), 384 kbps (class 2), and 128 kbps (class 1), corresponding to peer distributions 15%, 25%, 40%, and 20 % respectively. The reason for selecting this particular distribution is that they are mostly used in recent measurement studies on live streaming P2P systems [15][16][17].

To demonstrate the performance of DOA we form a randomly created overlay with 2000 nodes where $M_I=M_B=M_s=8$. In this paper we consider this variable constant and we will focus on its optimization in future work. Each node executes the DOAs every 1 sec and the service rate μ of the stream is equal to 90% (around 900 kbps) of the average uploading capacity.

In graph 1 we present the cumulative density function of the energy of each node divided by the number of its neighbors in the randomly formed overlay before the appliance and after converge of our algorithm. As we observe the mean energy (50^{th} percentile of the CDF) has been reduced from 0.07 to around 0.006 (more than 90%) which corroborates the locality properties of our overlay.

In graph 2 we demonstrate the speed of the reduction of the energies by executing again the same extreme scenario according to which we randomly insert initially 2000 peers and then we apply our algorithms. We observe that the mean energy (50^{th} percentile of the CDF) is reduced by a factor of 65% (from 0.07 to 0.023) in only 20 seconds, obviating the fast convergence properties of our optimization algorithm.

Using the same scenario, Graph 3 presents the CDF of the number of neighbors that each peer has after convergence of algorithms. Every slow peer (class 3,4) has exactly 16 neighbors, 8 neighbors in the base overlay and 8 interconnections to the super overlay. These interconnections are distributed to the class 1 and 2 peers in the super overlay according to their capacity. As the excess bandwidth of class 2 peers is minimal compared to class 1 peers the interconnections are distributed to these classes with a ration 31/1. So every class 2 peer has around 9 neighbors (8 in the super overlay plus one interconnection to the base overlay on average) while every class 1 peer has between 39 and 43 neighbors accordingly. Graph 3 shows that our algorithms eventually balance the number of neighbors that peers of the same class have.

In graphs 4 and 5 we show the performance of our system under static conditions. We simulated a system with 2000 nodes in which we applied our DOA until its convergence and then started the streaming process which takes place for 50 seconds (larger values have no effect on the results). We set the values of N_b and t_s to 14 blocks and 2 sec, respectively. The video streaming

playback rate μ is set to the 95% of the average capacity. Finally, as in static conditions the STTs between neighboring peers are very close to the minimum possible, we set the request interval to $2/N_b$ equal to 140 ms. We should note here that the value of the request interval has no correlation with the value of N_b, but as every peer sends its buffer to its neighbors $1/N_b$ seconds with this way we reduce the control overhead of our scheduler by piggy-packing the token and request messages within the buffer transmissions.

In Graph 4 we demonstrate the effectiveness of our scheduler by means of the CDF of successful block receptions from peers. Within a very small setup time period our system manages to successfully deliver a stream of a video with rate very close to the average uploading capacities of the peers. This exhibits the high degrees of bandwidth utilization that can be achieved with our algorithm.

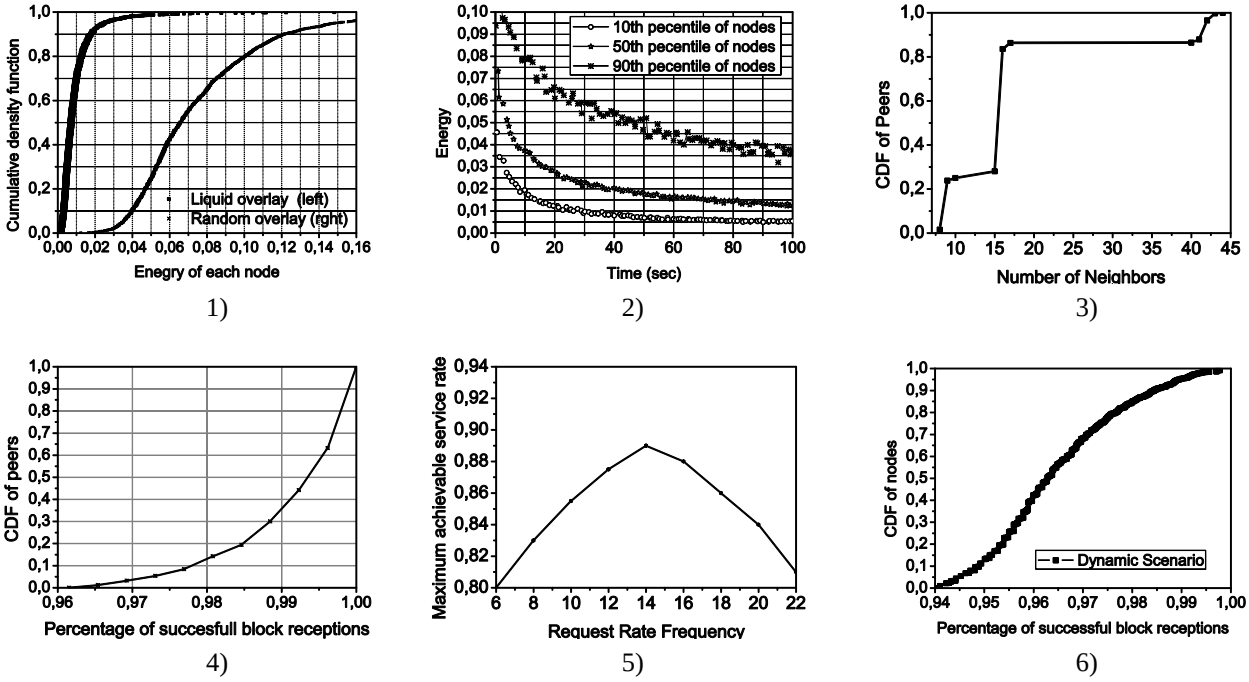

Figure 10: Graph 1-energy of peers, Graph 2-speed of energy minimization, Graph 3-number of neighbors, Graph 4-percentage of the successful block receptions of nodes in static conditions, Graph 5- request rate frequency, Graph 6- dynamic scenario

Graph 5 presents the video playback rate that the system is able to deliver as a function of request rate frequency. As we observe the performance is sensitive to request rate frequency. In more detail there is an optimal value that in the graph is around 14. The optimal request frequency is correlated with the energies of the participating peers. Our algorithms minimize these energies and so the setup time of our system can be decreased by a significant factor.

In Graph 6 we evaluate our system under dynamic conditions. Using the same initial conditions as in the previous cases, we simulate our system with 2000 nodes and N_b equal to 14 and the service rate equal to 90% of the available uploading capacity. In order to make the system more stable with respect to the presence of dynamic insertion and departures of peers, which has an impact on the energies of the participating peers, we have set the value of `request_interval` from $2/N_b$ to $3/N_b$. This increase has led to an analogous increase of the setup time from 2 seconds to 3 seconds so that content diffusion is accomplished before content downloading takes place.

Under these conditions, we evaluate our system under extreme dynamic conditions in terms of peer behavior and underlying network changes (graph 6), whereby 2000 nodes enter our system from 0 to 200 sec (10 peers/sec), whereas half of them depart during the period 150 to 250 sec (10 peers/sec). Also, the uploading bandwidth of each peer fluctuates dynamically every 2 seconds between -20% and +20% of its nominal value, according to a uniform distribution. As we can observe the degradation of our system performance is less than 3% compared to the static scenario. This shows the high tolerance of our system to dynamic conditions and also the very fast convergence of our optimization algorithm.

In the remaining four graphs we evaluate our QoS mechanism. We demonstrate a scenario in which 500 peers participate in the system. The video playback rate that is distributed is μ=900 kbps and it lasts for 300 seconds. In Graph 7 we demonstrate the average uploading capacity of the participating peers over time. This scenario covers all possible cases. As we can observe from the graph it includes cases where the average upload bandwidth of the participating peers is less than the required as well as cases where it is more. Additionally it includes discrete variations (steps) in order to make the problem more challenging.

In Graph 8 we demonstrate the percentage of the peers that received each block during the executed scenario. As we observe more than 99% of the peers manage to receive each block, something that testifies the stability achieved by our proposed algorithms and the continuous operation of our system independently of the average upload capacity of the participating peers.

In order to show that our proposed algorithms rely on the provision of excess bandwidth from the provisioning servers only when the aggregate peer resources are insufficient to sustain the video rate being delivered and, thus, manage to take full advantage of the peer resources, we depict in Graph 9 the average $T_{working}$ of the participating peers over time, where $T_{working}$ is defined as the percentage of time in which peers were transmitting blocks. As we can observe, the value of this variable is with high probability more than 95%, which means that the bandwidth of the provisioning servers is minimized while maintaining an uninterrupted service.

Finally, in Graph 10, we show the average upload bandwidth of the system over time, taking into account the upload bandwidth provided by provisioning servers as it is calculated by our proposed monitoring algorithm. We can observe that the average is always above the rate of the stream being delivered, and thus resulting in the uninterrupted streaming service. We can also observe that in time intervals where the uploading bandwidth of the peers changes simultaneously, our algorithms react fast and manages to keep the average bandwidth above the desired levels.

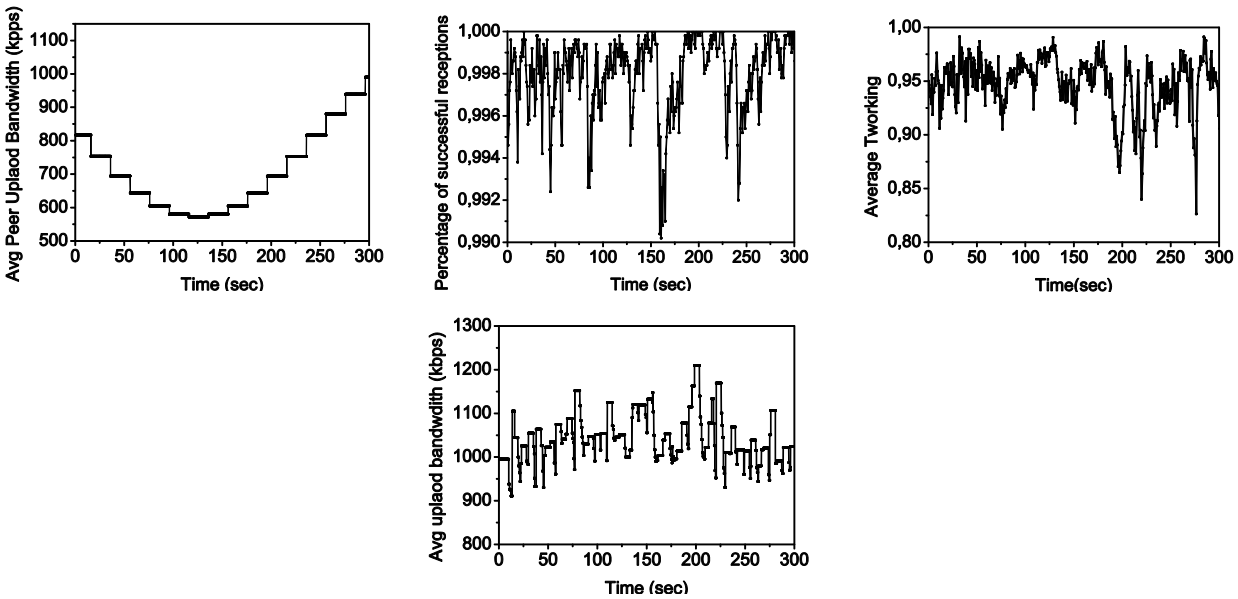

Figure 11. *Graph 7(upper left)-Average upload bandwidth of the participating peers, Graph 8-(upper middle), Percentage of peers that receive each block. Graph 9(upper right)- Average percentage of time that peers utilize their upload bandwidth, Graph 10(down)-Average upload bandwidth of the system*

VI. CONCLUSIONS

In this paper we have introduced a network architecture composed of signaling and data plane functions for the realization of live streaming applications using P2P content discovery and delivery methods. As it has been illustrated in the last section our algorithms for live streaming implementation has the capability of streaming content with the boundaries set by the user QoS settings.

This is made possible by achieving a service rate very close to the average available uploading bandwidth of the participating peers, while maintaining low set-up times even under dynamic network conditions. These results were achieved on an overlay architecture that takes into account locality information among peers to satisfy the downloading rate set by the user. Construction of this overlay system is underpinned by the two real-time optimization algorithms we have proposed, both of which are tolerant to the dynamic behavior of peers and the topology of the underlying network, while content distribution is greatly facilitated with the introduction of a new scheduler, which distributes the video stream among the peers and towards the user in a balanced way

ACKNOWLEDGEMENT

This work is funded from the ICT European project VITAL++ with Contract Number: INFSO-ICT-224287.

REFERENCES

[1] S. Guha, N. Daswani, and R. Jain, An Experimental Study of the Skype Peer-to-Peer VoIP System, in IPTPS'06, 2006.

[2] A. Cuevas, J.I. Moreno, P. Vidales, H. Einsiedler, The IMS service platform: a solution for next-generation network operators to be more than bit pipes. IEEE Communications Magazine, Vol. 44, Issue 8, pp. 75-81, 2006.

[3] S. L. Tompros, S. Denazis, Interworking of heterogeneous access networks and QoS provisioning via IP multimedia core networks. Computer Networks, Vol. 52, Issue 1, pp. 215-227, 2008.
[4] L. Kontothanassis, R. Sitaraman, J. Wein, D. Hong, R. Kleinberg, B. Mancuso, D. Shaw, D. Stodolsky, A transport layer for live streaming in a content delivery network, Proc. IEEE, vol. 92, no. 9, pp. 1408– 1419, 2004.
[5] K. Sripanidkulchai, B. Maggs, and H. Zhang, An analysis of live streaming workloads on the Internet, in ACM IMC, pp. 41–54, 2004.
[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Internet RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1, 1999.
[7] J. Rosenberg , H. Schulzrinne , G. Camarillo , A. Johnston , J. Peterson , R. Sparks , M. Handley , E. Schooler, SIP: Session Initiation Protocol, RFC Editor, 2002.
[8] F. M. Cuena-Acuna, C. Peery, R. P. Martin, T. D. Nguyen., PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. Rutgers Technical Report DCS-TR-487, 2002.
[9] F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris. Designing a DHT for low latency and high throughput. In Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI '04), 2004.
[10] Dong-Sung Kim, Jar-Min Lee, Wook Hyun Kwon, Design and Implementation of Home Network Systems Using UpnP Middleware for Networked Appliances, IEEE Transactions on Consumer Electronics, pp. 965 – 970, 2002.
[11] J. E. Bardram, The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications. Technical Report CfPC Technical Report 2004–PB–61, Centre for Pervasive Computing, Aarhus, Denmark, 2003. Available from http://www.pervasive.dk/publications.
[12] R. Jennings, E. Nahum, D. Olshefski, D. Saha, Zon-Yin Shae, C. Waters, A Study of Internet Instant Messaging and Chat Protocols, IEEE Network, pp 6-21, 2006.
[13] Antonio Liotta, Ling Lin, Managing P2P services via the IMS, 10th IFIP/IEEE International Symposium on Integrated Network Management, 2007.
[14] www.opnet.com.
[15] C. H. Ashwin R. Bharambe and V. N. Padmanabhan, Analyzing and Improving a BitTorrent Network Performance Mechanisms. IEEE INFOCOM, 2006
[16] Fabio Picconi and Laurent Massoulie, Is there a future for mesh-based live video streaming? IEEE P2P 2008.
[17] Ana Couto, Emilio Leonardi, Marco Mellia, Michela Meo, A Bandwidth-Aware Scheduling Strategy for P2P-TV Systems IEEE P2P 2008.
[18] http://www.ict-vitalpp.upatras.gr
[19] Athanasios Christakidis, Nikolaos Efthymiopoulos, Jens Fiedler, Shane Dempsey, Konstantinos Koutsopoulos, Spyros Denazis, Spyridwn Tombros, Stephen Garvey, Odysseas Koufopavlou, VITAL++ A New Communication Paradigm: Embedding P2P Technology in Next Generation Networks, IEEE Communications Magazine, 2011
[20] Nikolaos Efthymiopoulos, Athanasios Christakidis, Spyros Denazis, Odyssseas Koufopavlou, LiquidStream – Network dependent dynamic P2P live streaming, Springer, Peer-to-Peer networking and applications, vol. 1, no. 1, Jan. 2011
[21] Athanasios Christakidis, Nikolaos Efthymiopoulos, Spyros Denazis, Odyssseas Koufopavlou (2009) On the architecture and the design of P2P live streaming system schedulers, International conference on ultra modern telecommunications ICUMT 2009
[22] PPLive, http://www.pplive.com
[23] PPStream, http://www.ppstream.com
[24] SopCast, http://www.sopcast.org
[25] TVants, http://tvants.en.softonic.com
[26] Joost, http://www.joost.com
[27] D. Wu, Y. Liu, K.W. Ross, Queuing Network Models for Multi-Channel Live Streaming Systems, IEEE INFOCOM 2009
[28] Dan-Cristian Tomozei, Laurent Massoulie, Flow Control for Cost-Efficient Peer-to-Peer Streaming, INFOCOM 2010
[29] R. Kumar, Y. Liu, and K. W. Ross, Stochastic Fluid Theory for P2P Streaming Systems, IEEE INFOCOM 2007
[30] Nazanin Magharei, Reza Rejaie, PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming,IEEE INFOCOM , 2007
[31] Laurent Massoulie, Andy Twigg, Christos Gkantsidis, Pablo Rodriguez Randomized decentralized broadcasting algorithms. IEEE INFOCOM 2007
[32] Meng ZHANG, Qian ZHANG, Lifeng SUN, Shiqiang YANG, Understanding the Power of Pull-Based Streaming Protocol: Can We Do Better?, IEEE JSAC 2007
[33] Distributed Scheduling Scheme for Video Streaming over Multi-Channel Multi-Radio Multi-Hop Wireless Networks," IEEE Journal on Selected Areas in Communications, vol. 28, no. 3, pp. 409-419, Apr. 2010