QoS - LiquidStream: Scalable monitoring and bandwidth control in P2P live streaming

Nikolaos Efthymiopoulos, Athanasios Christakidis, Pierpaolo Giacomin, Spyros Denazis, Odysseas Koufopavlou
Electrical and Computer Engineering, University of Patras, Patras, Greece
{schristakidis, nefthymiop}@ece.upatras.gr, yrz@anche.no, {sdena, odysseas}@ece.upatras.gr

Abstract— **There is a lot of research work that has recently focused on P2P live streaming. As a result a lot of algorithms and system architectures have been proposed that achieve high upload bandwidth utilization of the participating peers, low delays during stream diffusion while systems are stable under dynamic network conditions and peer behavior. However, these systems make the assumption that the average upload bandwidth capability of the participating peers is stable and greater than the bandwidth the system requires for the successful delivery of the video stream, an assumption that makes these system impractical. This paper relaxes this assumption by proposing a monitoring mechanism for the total upload bandwidth capabilities of the participating peers, which is scalable, accurate, dynamic and with low overhead. Exploiting this monitoring mechanism, we also present and evaluate a method that allows the accurate and timely estimation of the required (minimal) amount of bandwidth that an external set of resources has to contribute in order to maintain uninterrupted stream delivery at the specified quality, thus making P2P systems practical for use.**

Index Terms— ***p2p live streaming, scalable bandwidth monitoring***

I. INTRODUCTION

P2P live streaming has been a thoroughly researched topic with many contributions from the scientific community [2][3][4][8][10][11][13][14], and with many commercial systems [16][17][18][19]. 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.

Achieving this, the research community has identified the following technical objectives:

1. The maximization of the utilization of the upload bandwidth of participating peers with heterogeneous upload bandwidth capabilities. The sum of these capabilities corresponds to the total amount of resources that P2P live streaming system must exploit. As stated in [3], at each time instant the average upload bandwidth of the participating peers forms an upper limit to the bit rate that any p2p system is able to deliver.
2. The minimization of the latency, called setup time, defined as the time interval that the system needs between the generation of a block in the "source" until its distribution to every participating peer.
3. The fast adaptation of the p2p system to user behavior. As peers enter and leave these systems dynamically, every successful system must be able to adapt its flows dynamically to the peers that contribute resources at each time instant.
4. Every P2P live streaming system must be able to distribute uniformly the sum of the upload bandwidth of participating peers in order to ensure that every peer will eventually acquire all video blocks in case of sufficient average upload bandwidth. If the average upload bandwidth becomes less than the bit rate of the stream, then an efficient system must deliver the same percentage of video blocks to every participating peer.

In contrast, ISPs are trying to find ways to minimize the cost of content distribution per user without compromising quality of the service. This is dependent on the number of video servers and caching centers that have to be deployed and maintained, the outgoing network bandwidth allocated to them for serving their customers, variable user numbers and demands thereof, etc. Using P2P as an alternative or complementary mechanism to mainstream content distribution it also introduces an additional objective which is the traffic minimization in the underlying network generated by the participating peers.

It is obvious that the objectives presented above are interrelated and interlinked creating various important trade-offs, which, in turn, have an impact on the way that P2P overlay architectures and p2p schedulers should be designed.

For instance, P2P overlays that manage to maintain high levels of upload bandwidth utilization reduce the amount of server resources that ISPs should commit for successful content delivery.

Fortunately, there are now effective overlay algorithms [8][2] that lead to upload bandwidth utilization more than 90%, while setup time in a live streaming system has been brought down to 5-10 sec or even less [11]. Finally, the most studied problem in P2P live streaming is the creation of systems that adapt the block transmissions very dynamically according to the state of the block reception that peers have each time instant. Recently proposed P2P block exchange scheduling architectures, as in [10], try to solve this problem.

Nevertheless, when these systems are deployed in real conditions as commercial or operational p2p live streaming systems [16][17][18][19] they fail to achieve the reported performance and stability. Even an inexperienced user can notice high latencies, instability and low bit rates of the delivered video stream.

This is mainly attributed to the fact that the average of the upload bandwidth of the participating peers changes unpredictably through time according to the set of the peers that participate in the stream distribution and the network conditions that might occur at each time instant. As a result, the designers of these systems act conservatively and select low stream bit rates for achieving high probability on delivery. This is a reason for low bit rates that users experience in commercial P2P live streaming systems. In the occasions when the average upload bandwidth of the peers falls below the bit rate of the stream, users experience instability and missed video blocks as no algorithm can deliver the stream under such circumstances.

Recently, the research community focused on video coding algorithms and information models that are able to adapt the required resources for the video distribution to the available bandwidth resources of the participating peers. However, these efforts still suffer from some of the limitations mentioned above as they trade quality for stability.

In this paper we focus on a complementary problem that we also consider very important. Thhis is the real time, dynamic, scalable and accurate estimation of the total resources that each P2P live streaming system offers. We also focus in the exact calculation of the resources that are missing or over provisioned. Finally we propose an architecture that allocates and embeds in the overlay efficiently auxiliary peers for the efficient and stable stream distribution. We highlight here the three major innovations of our work.

Firstly, we propose a novel scalable architecture able to monitoring in real time, dynamically and accurately the average upload bandwidth of participating peers in a video distribution by introducing negligible overhead to the system. The output of this algorithm is an estimation of the total bandwidth resources that each object distribution has. Such an estimation is accurate and on time.

Secondly, we propose an algorithm that uses as an input, the output of the monitoring algorithm. The objective of such an algorithm is the dynamic allocation of the minimal resources required from auxiliary peers towards the complete distribution of the video stream. This algorithm uses as input a set of nodes (peers and/or servers) that we note as auxiliary peers and their uploading capabilities, the amount of resources that normal peers have as the previous algorithm estimated and amount of resources that the distribution requires. The output of this algorithm is the amount of bandwidth that each auxiliary peer has to contribute by having as objective to have auxiliary peers that share the workload dynamically and according to their bandwidth capacity.

Thirdly, we propose how to extend this architecture in order to involve dynamically auxiliary peers in the system and exploit the allocated resources in a distributed and dynamic fashion. In order to do that we exploit the overlay presented in [2] which is an overlay adaptable to dynamic bandwidth changes, suitable for environments where nodes have high levels of heterogeneity in terms of their upload bandwidth capabilities and adaptable to the underlying network topology and traffic.

In this work we build on our previous work [2] by exploiting desirable properties of our overlay and scheduler which are necessary for the efficient functionality of the proposed monitoring and bandwidth control architecture. The motivation behind our work and more information related to the complete architecture of the deployment of content distribution services with the use of P2P architecture are described in [20].

The remaining of the paper is structured as follows. In section II we formulate the problem statement. In section III we describe our previous work that is used as background knowledge in this paper. In section IV we present our monitoring mechanism and its features. In section V we describe the way that upload bandwidth resources that system requires are estimated, and propose a policy for balanced resource allocation among auxiliary peers. In section VI we demonstrate how auxiliary peers are dynamically organized in the stream diffusion graph. In section VII the evaluation of our system is presented. Finally, section VIII presents the conclusions of this paper and the future work.

II. PROBLEM STATEMENT

Every p2p live streaming system involves at every time instant t, a set of participating peers, which we note as S(t) while each one of them contributes to the system his upload bandwidth, $c_i(t)$, $i \in S(t)$. We also assume that at any time instant there is another non empty set, say $S_{aux}(t)$, of *auxiliary* peers that can become available upon request by contributing certain amounts of upload bandwidth, $c_{aux,i}(t)$, $i \in S_{aux}(t)$.

These auxiliary nodes could be servers from a service provider, caches in the network or regular users that participate in other object distributions and have idle resources although they do not belong to S(t). Furthermore, they may be used to support various application scenarios such as server assisted p2p live streaming, multi-channel p2p live streaming and network cache assisted p2p live streaming.

As auxiliary resources cost, sharing them across a number of overlays in an optimal way is of paramount importance. To this end, minimizing the amount of auxiliary resources allocated to a specific object distribution without compromising stability and efficiency, maximizes sharing among different overlays. Assuming that a P2P live streaming system needs to perform at a delivery rate of μ bps, which is the video rate at the source (stream generator), and the bandwidth that an auxiliary node i can contribute is $c_{aux,,i}(t)$, at any time instant, we seek to minimize the following aggregate function that represents the total amount of upload bandwidth contributed by set $S_{aux}(t)$, namely,

$$min \sum_{i \in S_{aux}(t)} c_{aux,i}(t) \qquad (1)$$

We denote now *f(i,j)* as the upload rate of peer i directed to peer j. In order the whole P2P system to guarantee delivery of the stream to every peer the following equations must hold.

$$\sum_{i \in S(t) \cup S_{aux}(t)} f(i,j) \geq \mu, \forall j \in S(t) \cup S_{aux}(t) \quad (2)$$

$$\sum_{j \in S(t) \cup S_{aux}(t)} f(i,j) \leq c_i(t), \forall i \in S(t) \quad (3)$$

$$\sum_{j \in S(t) \cup S_{aux}(t)} f(i,j) \leq c_{aux,i}(t), \forall i \in S_{aux}(t) \quad (4)$$

Equation (2) represents the total incoming rate to a peer j while the sum of equations (3) and (4) represent the total outgoing rate from peer i. We assume that N is the cardinal number of $S(t) \cup S_{aux}(t)$. If we now sum up equation (2) for each participating peer (N equations), we end up with the total incoming flow that the system generates. In contrast, if we sum up equations (3) and (4) for the normal and auxiliary peers respectively, we end up with the total outgoing flow that the system generates (N equations). In every p2p live streaming system the outgoing flow is equal with the incoming flow. So if we subtract the N equations of type (3) and (4) from the N equations of type (2) we have:

$$\sum_{i \in S_{aux}(t)} c_{aux,i}(t) \geq N * \mu - \sum_{i \in S(t)} c_i(t) \qquad (5)$$

Examining equations (1) and (5), the equality in (5) corresponds to the minimal amount of bandwidth that auxiliary peers have to contribute for the successful delivery of the stream.

In order to calculate and allocate the amount of bandwidth that each auxiliary peer has to contribute, we need to accurately estimate the total amount of upload bandwidth resources contributed by the participating peers, S(t). The periodic measurement of the upload capacity of every participating peer is a process that it is simply not scalable while it is very difficult to measure the upload bandwidth in such a way that readily captures bandwidth fluctuations. In order to meet these requirements (scalability, accuracy and responsiveness) there is a need for a measurement process that relies on statistical sampling. In section III we describe such a process that estimates the total upload bandwidth capabilities of the participating peers by exploiting attributes that p2p block schedulers have.

Having estimated the total contributing upload bandwidth by the participating peers, we use equation (5) in order to calculate the total upload bandwidth that auxiliary peers have to contribute. We, then, apply a bandwidth allocation algorithm that apportions this bandwidth to the set of auxiliary peers that belong to $|S_{aux}(t)|$ in order to have equal percentages of bandwidth utilization. The bandwidth allocation algorithm may result in two possible outcomes. The first is the allocation of more bandwidth through auxiliary peers when the average uploading capacity of the participating peers decrease, whereas the second is the reduction of the bandwidth that the set of auxiliary peers already contribute in case where the average upload capacity of the participating peers belonging to S(t) increase.

Finally, there is a need for an efficient combination of a p2p overlay algorithm and scheduler that readily react to arrivals and departures of (auxiliary) peers and adapt to resource changes. Such a system is briefly presented in the next section.

III. P2P LIVE STREAMING

In [2] we focus on the development of a live streaming system that consists of two inter-related entities: the content diffusion overlay (CDO) and the P2P block exchange scheduler (BESA).

1. Content Diffusion Overlay

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 (Figure 1).

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 it 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. If this peer also happens to be a super peer then it is also admitted to an additional 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.

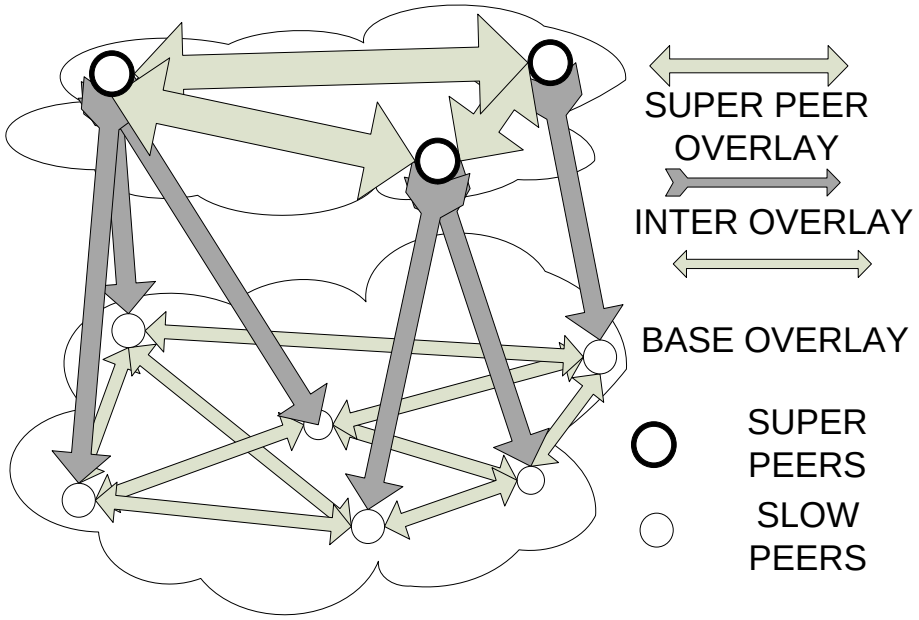


FIGURE 1. The structure of the overlay grph.

Our *inter 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 1 depicts such a system overlay with $M_B=3$, $M_S=2$, and $M_I=1$.

All the peers periodically execute a distributed optimization and maintenance algorithm (DOMA) that reorganizes the "neighborhoods" of CDO in order to keep the structure of the graph optimal for content delivery even during peer arrivals and departures. The optimization algorithm makes use of an "energy function" that captures the impact of specific parameters such as network latency among peers thus creating locality aware overlays.

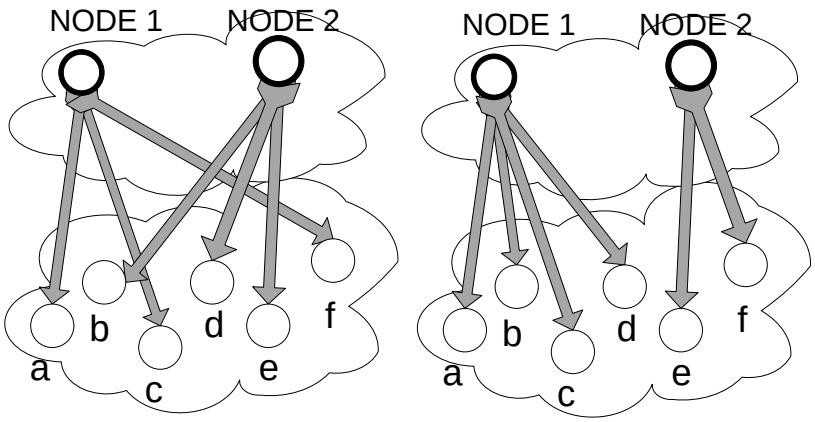


FIGURE 2. Execution of DOMA

Any change in the underlying network conditions, in the amount of resources contributed by a peer, in peer arrivals and departures triggers a DOMA execution. We stress here that DOMA always converges to the desired graph structure and to a minimized sum of energies [2].

2. P2P block exchange scheduling algorithm

In conjunction with DOMA running in peers, a P2P block exchange scheduler algorithm (BESA) ensures the distribution of each block to every peer in the CDO. This scheduler has been designed in such a way that exploits the structure and properties e.g. locality, of CDO, and is executed periodically by each peer. At each time interval, every peer acts as a sender and receiver simultaneously using the corresponding BESA decision functions.

In order 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 exactly before the beginning of the transmission of a block. On the other hand the receiving peer proactively notifies candidate sending peers about the block that it wishes to receive. In this way, we can achieve fast and complete diffusion of each block while we avoid duplicate block transmissions from different sending peers to the same receiving peer. We consider this receiver driven block selection approach as the most efficient one in distributing blocks since the receiving peer has always 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 has been shifted toward coordinating these sending peers in a distributed manner such that they avoid duplicate block transmissions and prioritize the transmission of rare blocks in a neighborhood.

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

BESA achieves high bandwidth utilization, fast diffusion of each block, uniform distribution of the total upload bandwidth resources to the participating peers and low control overhead. We exploit these properties towards the development of our monitoring and resource allocation system.

IV. P2P MEASUREMENTS

This section describes a measurement process that constantly estimates the overall upload bandwidth capabilities of a P2P system. It uses only a small subset of the participating peers which measure periodically their incoming bit flows during a time period, say T_{Δ}. These measurements are then reported to a server that periodically executes an algorithm in order to estimate the total upload bandwidth of the system.

During every time interval T_{Δ}, each peer i has been either busy transmitting for a period $T_{busy}(i)$, or idle for a period $T_{idle}(i)$, thus $T_{\Delta}=T_{busy}(i)+T_{idle}(i)$, $\forall i$. Accordingly, every peer i could estimate its upload bandwidth $c_i(T_{\Delta})$ as the ratio of the transmitted bits, $B_{out}(i)$, over the time interval $T_{busy}(i)$. Summing up all measurements carried out by peers we can express the total upload bandwidth of every set of peers, namely,

$$\sum_{\forall i} c_i(T_{\Delta}) = \sum_{\forall i} \left[\frac{B_{out}(i)}{T_{busy}(i)}\right] \qquad (6)$$

or, alternatively,

$$\sum_{\forall i} c_i(T_{\Delta}) = \sum_{\forall i} \left[\frac{B_{out}(i)}{T_{\Delta}-T_{idle}(i)}\right] \qquad (7)$$

In order to simplify the sum in equation (7) we will exploit an attribute of the proposed BESA as described earlier. This attribute is that time interval $T_{idle}(i)$ is almost the same in every participating peer. More precisely we executed off line the Kolmogorov – Smirnov test [7] in the values of this metric among all participating peers and we testified that it approximates a normal distribution.

In order to be able to calculate the average values of the parameters above we resort to sampling by selecting a small subset of peers, say S_{sample} making sure that the selected size of the subset gives accurate estimates of the averages.

Let us assume at the moment that a measurement server has already selected the member peers of such a set S_{sample} and requests from them the values of $T_{idle}(i)$ measured during the last period T_{Δ}. The statistical average $E[T_{idle}]$ and the statistical standard deviation σ_{Tidle} can be calculated according to the following equations:

$$E[T_{idle}] = \frac{\sum_{i \in S_{sample}} T_{idle}(i)}{|S_{sample}|} \qquad (8)$$

$$\sigma_{T_{idle}} = \sqrt{\sum_{i=1}^{|S_{sample}|} \frac{(T_{idle}(i)-\overline{T_{idle}})^2}{|S_{sample}|-1}} \qquad (9)$$

With probability α, the value $T_{idle}(i)$ of a participating peer i lies in the following interval:

$$E[T_{idle}] \mp z_{a/2} * \sigma_{T_{idle}} / \sqrt{|S_{sample}|} \qquad (10)$$

Where $z_{\alpha/2}$ is the value of a random variable that follows a normal distribution for which the integral of the probability density function of the normal distribution is α%. We highlight here two facts. The first is the very low statistical standard deviation of this metric as our BESA tends to using in the upload bandwidth of the participating peers with the same percentage. The second is that the value of T_{Δ} is much higher than $T_{idle}(i)$. As we explained in the previous section this is due to the fact that we want to minimize the upload bandwidth that auxiliary peers contribute and the operation of the system, i.e $T_{idle}(i) \ll T\Delta = T_{busy}(i) + T_{idle}(i)$. According to these we introduce an error smaller than

$$z_{a/2} * \sigma_{T_{idle}} / \sqrt{|S_{sample}|} \qquad (11)$$

So with high accuracy according to equation (11) we can express equation (7) as:

$$\sum_{\forall i} c_i(T_{\Delta}) = \frac{\sum_{\forall i} B_{out}(i)}{T_{\Delta}-\overline{T_{idle}}} \qquad (12)$$

In every p2p system, at a given time interval, the sum of the flows that peers upload is identical with the sum of the flows that peers download. Roughly speaking the total amount of outgoing bits is identical with the total amount of incoming bits. According to this fact we can rewrite equation (12) as:

$$\sum_{\forall i} c_i(T_\Delta) = \frac{\sum_{\forall i} B_{in}(i)}{T_\Delta - \overline{T_{idle}}} \qquad (13)$$

The statistical average $E[B_{in}]$ and the statistical standard deviation $\sigma_{B_{in}}$ can be calculated again according to the following equations:

$$E[B_{in}] = \frac{\sum_{i \in S_{sample}} B_{in}(i)}{|S_{sample}|} \qquad (14)$$

$$\sigma_{B_{in}} = \sqrt{\sum_{i=1}^{|S_{sample}|} \frac{(B_{in}(i) - \overline{B_{in}})^2}{|S_{sample}| - 1}} \qquad (15)$$

By applying again to these measurements the Kolmogorov – Smirnov test we again evaluate that the probability density function of various $B_{in}(i)$ approximates the normal distribution. Then again by modeling this distribution as the normal distribution we can estimate the average value of B_{in} with a confidence interval α according to:

$$P\left[-z_{\frac{a}{2}} \leq \frac{E[B_{in}] - \sigma_{B_{in}}}{\frac{\sigma_{B_{in}}}{\sqrt{|S_{sample}|}}} \leq z_{\frac{a}{2}}\right] = 1 - a \qquad (16)$$

Eq. 16 states that for every parameter that follows the normal distribution the mean value is estimated in the following interval with probability α,:

$$E[B_{in}] \mp z_{a/2} * \sigma_{B_{in}} / \sqrt{|S_{sample}|} \qquad (17)$$

If we set the parameter α very close to 100% (ex. α=0.99) we can estimate with very high confidence and by measuring only a small subset of peers that a safe estimation of the lower threshold of value $\overline{B_{in}}$ is:

$$\overline{B_{in}} = E[B_{in}] - z_{a/2} * \sigma_{B_{in}} / \sqrt{|S_{sample}|} \qquad (18)$$

As we observe from Eq. 18 the size of the interval in which we can estimate $\overline{B_{in}}$ depends on the size of set S_{sample} and on the statistical standard deviation $\sigma_{B_{in}}$. This observation has a great impact on the statistical behavior that should be sought after from BESA. The lower the statistical standard deviation the smaller the size of S_{sample} that is required in order to estimate and calculate $\overline{B_{in}}$. To this end, our BESA that achieves uniform block distribution among peers results in small sample sizes. We analyze this in detail later in the evaluation section. Exceeding a certain threshold has no effect in the accuracy of the measurements. It is this observation that our measurement algorithm exploits in order to achieve scalability and accuracy.

The parameters in the right part of equation (7) depend on index i (peer i). However, we are interested in average values for the purposes of our measurement process. Taking therefore the corresponding statistical averages $\overline{B_{in}}$ and $\overline{T_{idle}}$ for every peer i, we arrive at the following important equation that describes the overall uploading capability of a system comprised of a set of participating peers of size N.

$$\sum_{i=1}^{N} c_i(t) = \frac{N \times \overline{B_{in}}}{T_\Delta - \overline{T_{idle}}} \qquad (19)$$

As stated in Eq. 2 the total amount of flows that each peer i has to acquire from the participating peers for the complete reception of the stream is the video playback rate μ. At this point we want to clarify that p2p block schedulers introduce a control overhead during their operation due to network protocol headers, duplicate block transmissions, control messages exchanged among peers etc.

This overhead ratio is denoted as r_{OHD}. Each peer can easily measure at each T_Δ its total incoming amount of bits that is $\overline{B_{in}}$ and the amount of bits that are parts of video blocks that we note as $\overline{B_{in,actual}}$. Our measurement process

can also estimate this overhead r_{OHD} using the same peers in S_{sample}. To this end, at every period T_Δ we can also have an accurate estimate of this ratio as:

$$r_{OHD} = \frac{\overline{B_{in}}}{\overline{B_{in,actual}}} \quad (20)$$

V. BANDWIDTH ESTIMATION AND RESOURCE ALLOCATION

In this section we analyze how we can estimate the additional bandwidth that auxiliary peers have to contribute dynamically by using the measurements of the monitoring algorithm as an input. In addition, we propose an algorithm that calculates the amount of bandwidth that each auxiliary peer has to contribute in order to balance the percentage of the available bandwidth utilization among auxiliary peers. With this policy we avoid overloading a subset of auxiliary peers while we balance the percentage of resources that are left idle in each auxiliary peer. Accordingly, this amount of resources can be shared among different overlays e.g. multiple channel video distribution.

More specifically we note as DU_{req} the difference between the minimum required bandwidth in the system and the total available bandwidth that the system has. So we can now rewrite Eq. 5 as:

$$DU_{req} = N * \mu - \sum_{i \in S(t) \cup Saux(t)} c(i) \quad (21)$$

Towards the formulation of this equation in section II we assumed for the sake of clarity that p2p block schedulers introduce no overhead and so the total bandwidth that is required towards the complete distribution of a stream with bit rate μ in N participating nodes was N*μ. If we now consider the overhead of the p2p block exchange scheduling algorithms as analyzed in the previous section we can have a better estimation of the total required bandwidth. Furthermore the set $S(t) \cup Saux(t)$ is the set with the participating normal and auxiliary peers N. After these observations we can rewrite Eq. 15 as:

$$DU_{req,actual} = N * \mu - \sum_{i \in N} c(i)_{actual} \quad (22)$$

Where $DU_{req,actual}$ is the difference in the actual upload bandwidth that auxiliary peers have to contribute and $\sum_{i \in N} c(i)_{actual}$ is the actual capacity of the participating peers. We highlight again that with the term actual we note the bandwidth that is really useful if we exclude from the total bandwidth the overhead. This overhead is caused by a number of factors [2], among them are the network overhead (packet headers) and the overhead that p2p-BESA introduces. According to these the ratio between the bandwidth and the actual bandwidth with no loss of generality is equal to the overhead ratio. So we can rewrite equation (22) as:

$$DU_{req} = \left(N * \mu - \frac{\sum_{i \in N} c(i)}{r_{OHD}}\right) * r_{OHD} \quad (23)$$

By also using equation (19) and (20) we can rewrite (22) as:

$$DU_{req} = N * \left(\frac{\overline{B_{in}}}{\overline{B_{in,actual}}} * \mu - \frac{\overline{B_{in}}}{T_\Delta - \overline{T_{idle}}}\right) \quad (24)$$

As we observe from Eq. 23 in order to estimate the increase or decrease in the amount of bandwidth that auxiliary peers needs to contribute only the estimation of $\overline{B_{in}}$, $\overline{B_{in,actual}}$ and $\overline{T_{idle}}$ is needed. As we explained in the previous section these metrics have been selected for such estimation due to the fact that they have very low variance and so by measuring their values in a small subset of participating peers we can estimate with accuracy their total average in the whole system.

After the estimation of DU_{req} there are two cases that we may face. In the first one DU_{req} is positive and auxiliary peers have to contribute more upload bandwidth in order to have a successful delivery of the stream while in the second DU_{req} is negative and we have to lower the total upload bandwidth that auxiliary peers contribute in order to avoid the waste of network resources. This bandwidth can be used for other purposes (another object distribution). By the use of such an algorithm we obtain two advantages. The first is that we contribute the minimal amount of bandwidth towards the complete distribution of a stream. The second is that we have a system that is able to fulfill dynamically the bandwidth requirements by adapting the behavior of the auxiliary peers to the available bandwidth resources.

With this fast, accurate, scalable and with low overhead technique we obtain the amount of bandwidth that auxiliary peers have to contribute the next problem that we have to face is how we can distribute DU_{req} among the $S_{aux}(t)$

auxiliary peers that are available. We highlight here that our system does not do any assumption on the size and the amount of bandwidth that each helper contributes and so it is useful in any p2p live streaming architecture.

The objective in the sharing of the DU_{req} among auxiliary peers is the balanced utilization of their uploading capabilities. We model here this problem as a convex network optimization problem and we refer to [5] for its analytical algorithmic solution. We will now describe in detail the first case where DU_{req} is positive and so we need from the auxiliary peers to contribute more upload bandwidth while the second case where DU_{req} is negative and we require the degradation of the upload bandwidth that they contribute.

We note as $c_{aux}(i)$ the total uploading bandwidth capacity of each auxiliary peer i that belongs to $S_{aux}(t)$ and with $c_{auxi}(t)$ the upload bandwidth that i currently contributes and with $c_{auxi}(t+1)$ the upload bandwidth that i will contribute after the execution of the bandwidth allocation algorithm We can achieve to balance the utilization of the upload bandwidth of each auxiliary peer with equal percentage by modeling the reassignment of the upload bandwidth that auxiliary peers have to contribute as:

$$\min \sum_{i \in Sh(t)} \left(\frac{c_{hi}(t+1)}{c(i)}\right)^2 \qquad (24)$$

We introduce here a constraint where the upload bandwidth that the bandwidth allocation algorithm allocates to each auxiliary peer is less than its upload capacity. So for each auxiliary peer i the constraint is:

$$c_{auxi}(t+1) \leq c_{aux}(i) \qquad (25)$$

Finally the supplement of bandwidth that $S_{aux}(t)$ will contribute will be:

$$\sum_{i \in S_{aux}(t)} [c_{auxi}(t+1) - c_{auxi}(t)] = DUreq \qquad (26)$$

In case that the problem is infeasible means that the sum of the upload bandwidth that remains in $S_{aux}(t)$ is not enough and the only solution is the reduction of the bit rate of the video stream by using video coding techniques or a lower quality stream. We leave the solution of this problem as future work. In this work we consider that always the total amount of bandwidth that is available from $S_{aux}(t)$ is enough for the delivery of the multimedia streams.

We will now focus on the second case where DU_{req} is negative and so we have the opportunity to save upload bandwidth from the various auxiliary peers. We again want to:

$$\min \sum_{i \in S_{aux}(t)} \left(\frac{c_{auxi}(t+1)}{c_{aux}(i)}\right)^2 \qquad (27)$$

Under the constraint that:

$$\sum_{i \in s_{aux}(t)} [c_{auxi}(t+1) - c_{auxi}(t)] = DUreq \qquad (28)$$

But in this case we desire for each auxiliary peer:

$$c_{auxi}(t+1) \geq 0 \qquad (29)$$

In case where from these equations we obtain an infeasible problem auxiliary peers don't provide upload bandwidth.

VI. Overlay architecture

The set of the auxiliary peers that participate in the system is dynamic while their contributed upload bandwidth is continuously changing according to the output of the resource allocation algorithm. This fact creates the necessity of the continuous CDO adaptation to these changes. For this reason we insert these peers as super peers in the super peer overlay in order to exploit our DOMA towards the utilization of the excess bandwidth that they provide. The end result is the full exploitation of their upload bandwidth and the minimization of network traffic as these peers are connected to peers that are physically close to them in the underlying network. We again refer to [2] for the detailed description of DOMA.

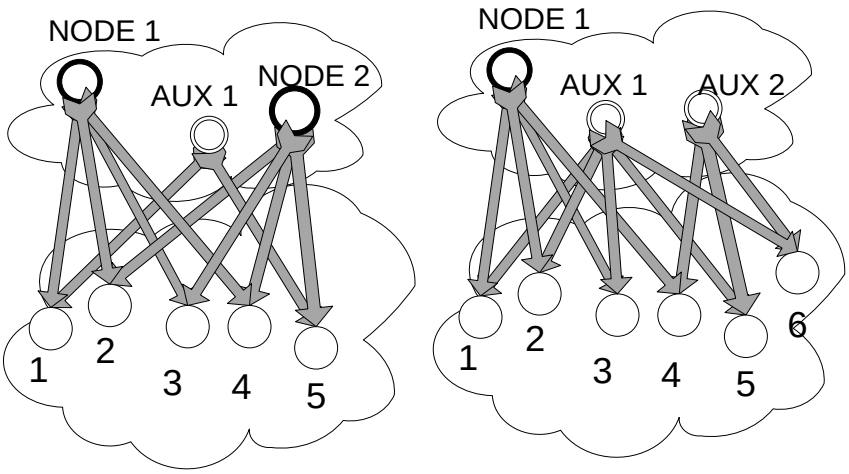


Figure 2. CDO adaptation to the output of resource allocation algorithm.

In the following figure we present an example of a part of the CDO in two cases. In the first case (left) there are two super peers with equal upload bandwidth C and an auxiliary peer that provides C/2 bandwidth. In the second case (right) one of the super peers has left the system, a new auxiliary peer has entered the system and the resource allocation algorithm modified the bandwidth that auxiliary peers contribute to ¾*C.

We highlight here that our distributed optimization algorithms ensure the distribution of the connections between the two overlays, to the super peers according to their upload bandwidth. Additionally these connections are distributed to slow peers according to the network latency between super peers and them. In the figure the length of the arc is proportional to the network latency among the connected peers.

VII. EVALUATION

For the evaluation of our P2P streaming system we have used the OPNET Modeler v.14 [6] in order to test our proposed system under various scenarios and conditions. Here we present its performance based on a network topology taken from [9]. In order to model heterogeneous uploading bandwidth capabilities of the participating peers we set the uploading bandwidths equal to 4000 kbps (class 4), 1000 kbps (class 3), 384 kbps (class 2), and 128 kbps (class 1). These uploading capacities have been observed in today's p2p systems.

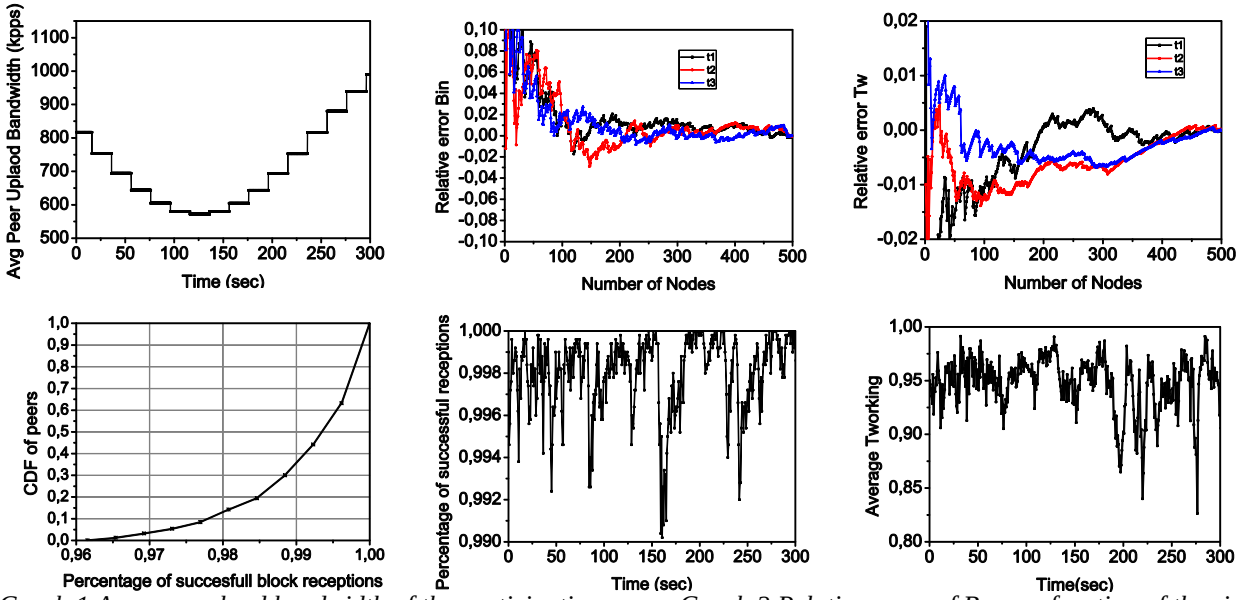
In this paper we demonstrate a scenario in which 500 peers participate in the system. The video playback rate that is distributed is μ=900 kbps and it lasts for 300 seconds. In Graph 1 we demonstrate the average uploading capacity of the participating peers over time. This scenario covers all possible 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 order to demonstrate the accurate estimation of $\overline{B_{in}}$ by sampling a small number of peers we gathered the value of $\overline{B_{in}}$ from every participating peer in 20 random time instances. Here we present the results form three different time instances (t1, t2 and t3) in which the values of the measured B_{in} of the peers exhibits the greatest variation. We show here how the estimation of $\overline{B_{in}}$ converges as the size of the set of the peers that we sample increases. In more detail in graph 2 we demonstrate the relative error in the estimation of $\overline{B_{in}}$ over different sampling sizes. This can be calculated from the following equation:

$$RE(\overline{B_{in}(sample)} = (\frac{\overline{B_{in}(sample)}-\overline{B_{in}(all)}}{\overline{B_{in(all)}}}) \qquad (31)$$

Where $\overline{B_{in}(sample)}$ is the estimation of the average value of |sample| measurements (peers) and $\overline{B_{in(all)}}$ is the real average of B_{in} of every participating peer. In graph 2 we demonstrate this estimation as a function of the size of the set of the peers that we embed in the estimation.

As it can be observed from graph 2, even in the worst of the cases the relative error stays below 10% even if we use only 10 peers, it further decreases to 5% if we increase the size of the peers that we sample to 50, and finally, it becomes just 2% if the size of the set to 100. We highlight here that the theoretical proof of the independence of the accuracy of our estimation algorithm form the size of the participating peers is also confirmed by our simulations.



Graph 1 Average upload bandwidth of the participating peers. Graph 2 Relative error of B_{in} as a function of the size of the set of the peers that we sample. Graph 3 Relative error of T_w as a function of the size of the set of the peers that we sample. Graph 4 Cumulative density function of peers of the successful block receptions. Graph 5. Percentage of peers that receive each block. Graph 6. Average percentage of time that peers utilize their upload bandwidth

This fact testifies the outstanding scalability properties of our monitoring algorithm, as it can be applied to systems of content distribution, in which thousands of peers participate, without reducing its accuracy.

Graph 3 depicts the relative error of the estimation of T_{busy}, over different sizes of set of peers which were sampled. We present here, again, the values of the relative error during the three different time instances, in which the convergence of the relative error was the slowest. As it can be observed from the graph, with a set size of just 50 peers, we can have a very accurate estimation of T_{busy} with a relative error of less than 2%.

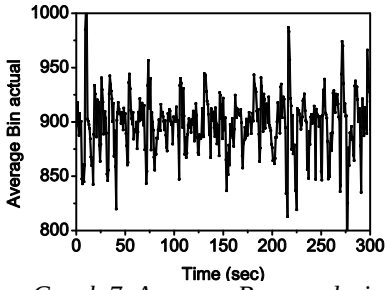
Graph 4 depicts the performance of the system using our proposed algorithm. The sample size is 50 peers with a sampling period of 1 second. In graph 5 we present the cumulative density function of the percentage of the successful block receptions of the participating peers. As we observe the mean value is more than 99% and even the peer which experienced the worst performance succeeded in acquiring more than 96% of the video blocks. These results could be better, meaning that every peer would have acquired the whole stream, by providing 2% more bandwidth through the auxiliary peers. We choose to demonstrate here a scenario where the auxiliary peers allocate bandwidth according to the aforementioned algorithms and equations in order to demonstrate their accuracy in the estimation of the required resources.

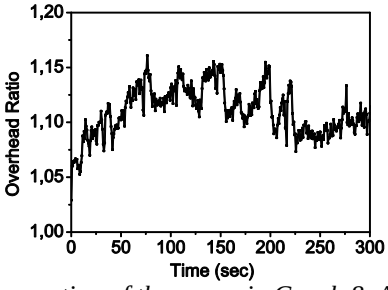
In Graph 5 we demonstrate the percentage of the peers that received each block during the executed scenario. As we observe more than 99% of the peers manage to receive each block, something that testifies the stability achieved by our proposed algorithms and the continuous operation of our system independently of the average upload capacity of the participating peers.

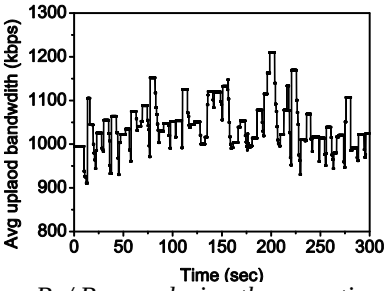
In order to show that our proposed algorithms rely on the provision of excess bandwidth from the auxiliary peers only when the aggregate peer resources are insufficient to sustain the video rate being delivered and, thus, manage to take full advantage of the peer resources, we depict in Graph 6 the average $T_{working}$ of the participating peers, over time. As we can observe, the value of this variable is with high probability more than 95%, which means that the bandwidth of the auxiliary peers is minimized while maintaining an uninterrupted service.

In Graph 7 we demonstrate the average $B_{in,actual}$ of the participating peers over time. As we observe $B_{in,actual}$ remains stable around 900 kbps, which is the value of the video playback rate. This testifies the ability of our proposed system to offer the amount of bandwidth that is required for the efficient, continuous and complete video distribution.

In Graph 8 we demonstrate the ratio between B_{in} and $B_{in,actual}$, as it is measured through the execution of the scenario, which is equal to the overhead that is introduced by the scheduler of the system. This overhead could be the







Graph 7. Average B_{in_actual} during the execution of the scenario Graph 8. Average B_{in}/ $B_{in,actual}$ during the execution of the scenario. Graph 9, Average upload bandwidth of the system

bandwidth needed for the transmission of the control messages, the bandwidth wasted in the transmission of duplicate packets etc. The purpose of this graph is to show how this ratio varies over time according to the bandwidth fluctuations.

Finally, in Graph 9, we show the average upload bandwidth of the system over time, taking into account the upload bandwidth provided by the auxiliary nodes as it is calculated by our proposed algorithms. 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.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper we demonstrate that exploiting scheduling fairness in a peer to peer live streaming system enables the estimation of the performance of such a system with low overhead and high accuracy. This makes a fair scheduler a prerequisite for any future peer to peer live streaming system.

Furthermore, we presented and validated an algorithm enabling the calculation of the discrepancy between the existing aggregate bandwidth of the system and the required uploading bandwidth in order to sustain an uninterrupted service.

We also combined this algorithm with an architecture where, in case that there is a need for a supplement for upload bandwidth, there is set of auxiliary peers that provide dynamically the exact amount that is required. Again we observed that our upload bandwidth allocation algorithm distributes the upload bandwidth to the auxiliary peers according to their load.

Finally, by taking advantage of the properties of our proposed overlay, we ensure the efficient utilization of the bandwidth provide by the auxiliary peers

As future work we will focus on the time multiplexing of peers upload bandwidth between different video distributions in order to exploit idle upload bandwidth in video distributions where the average upload bandwidth of the participating peers exceeds the bit rate of the video stream.

ACKNOWLEDGEMENT

This work is funded from the ICT European project VITAL++ [1] with Contract Number: INFSO-ICT-224287.

REFERENCES

[1] http://www.ict-vitalpp.upatras.gr/publications.html

[2] Efthymiopoulos Nikolaos, Christakidis Athanasios, Denazis Spyros , Odysseas Koufopavlou, LiquidStream – Network dependent dynamic P2P live streaming, Springer, Peer-to-Peer networking and applications (to be published and at the moment can be found in [1])

[3] R. Kumar, Y. Liu, and K. W. Ross, Stochastic Fluid Theory for P2P Streaming Systems, IEEE INFOCOM 2007

[4] Nazanin Magharei, Reza Rejaie, PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming,IEEE INFOCOM , 2007

[5] Dimirti P. Bertskeas, Network Optimization: Continuous and Discrete Models, Athena Scientific, May 1998
[6] www.opnet.com
[7] http://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test
[8] Meng Zhang, Qian Zhang, Lifeng Sun, Shiqiang Yang, Understanding the Power of Pull-Based Streaming Protocol: Can We Do Better?, IEEE JSAC 2007
[9] C. H. Ashwin R. Bharambe and V. N. Padmanabhan, Analyzing and Improving a BitTorrent Network Performance Mechanisms. IEEE INFOCOM, 2006
[10] Fabio Picconi and Laurent Massoulie, Is there a future for mesh-based live video streaming? IEEE P2P 2008
[11] Ana Couto, Emilio Leonardi, Marco Mellia, Michela Meo, A Bandwidth-Aware Scheduling Strategy for P2P-TV Systems IEEE P2P 2008
[12] Svante Ekelin, Martin Nilsson, Erik Hartikainen, Andreas Johnsson, Jan-Erik Mång, Bob Melander, and Mats Björkman, Real-Time Measurement of End-to-End Available Bandwidth using Kalman Filtering, 10thIEEE/IFIP Network Operations and Management Symposium. NOMS 2006.
[13] D. Wu, Y. Liu, K.W. Ross, Queuing Network Models for Multi-Channel Live Streaming Systems, IEEE INFOCOM 2009
[14] Dan-Cristian Tomozei, Laurent Massoulie, Flow Control for Cost-Efficient Peer-to-Peer Streaming, INFOCOM 2010
[15] Dimitri Bertsekas, Linear Network Optimization: Algorithms and Codes, MIT Press, 1991
[16] PPLive, http://www.pplive.com
[17] PPStream, http://www.ppstream.com
[18] SopCast, http://www.sopcast.org
[19] TVants, http://tvants.en.softonic.com
[20] Athanasios Christakidis, Nikolaos Efthymiopoulos, Jens Fiedler, Konstantinos Koutsopoulos, Spyros Denazis, Spyridwn Tombros, Shane Dempsey, Odysseas Koufopavlou, VITAL++ A New Communication Paradigm Embedding P2P Technology in Next Generation Networks, IEEE Communications magazine 2011(to be published and at the moment can be found in [1])