LiquidStream – Network dependent dynamic P2P live streaming

Nikolaos Efthymiopoulos, Athanasios Christakidis , Spyros Denazis, Odysseas Koufopavlou
Department of Electrical and Computer Engineering,University of Patras, Greece
{nefthymiop, schristakidis, sdena, odysseas}@ece.upatras.gr

***Abstract*-A successful P2P live streaming system must achieve high uploading bandwidth utilization, fast stream distribution, uniform bandwidth distribution among participating peers, flexibility and adaptation to the underlying network conditions and peer behavior. This paper proposes a novel architecture that meets these requirements. By the use of distributed optimization algorithms we propose a dynamically reconfigurable overlay architecture that organizes its peers according to network locality information and heterogeneous uploading capabilities of them. The benefits of our optimized overlay are fully exploited by our proposed scheduler, which guarantees the complete and fast distribution of the stream. The evaluation of our system under a series of scenarios that take into account the all requirements above reveals the advantages of our proposed system.**

Keywords – P2P live streaming, distributed optimization

1. Introduction

Peer-to-Peer (P2P) live streaming is a heavily researched topic that faces a series of challenges originating both from the diversity of behaviors and capabilities of the participating peers combined with the strict delivery requirements of streaming applications. In the rest of this paper we assume a cooperative environment as [1][9][11][12] that could be possibly applied in set top boxes or another environment with the same feature.

In more detail, peers involved in these systems have heterogeneous and dynamic uploading bandwidth capabilities, which when combined with the characteristics of the topology of the underlying network and the dynamic traffic conditions e.g. latency, they create a dynamic and complex environment for P2P live streaming delivery. These factors heavily influence the efficiency of a P2P system measured by a number of key performance indicators.

The first factor is *the uploading bandwidth utilization* that corresponds to the ability of the system to exploit as much as possible the sum of the uploading bandwidth of the participating peers that is noted as aggregate uploading bandwidth of the system.

Equally important is the *setup time* defined as the time interval between the generation of a block from the origin server and its delivery to the every peer in the system.

Furthermore, a P2P live streaming system has to remain *stable* in such an environment especially in the presence of frequent peer arrivals and departures. This results in varying numbers of peers which impact the stability of the system with respect to the uninterrupted delivery of the streaming service.

Finally, *fairness* among nodes indicates the ability of the system to continuously distribute uniformly the aggregate uploading bandwidth to the participating peers. This ability ensures that even in conditions where aggregate uploading bandwidth is insufficient for the delivery of the whole video stream all peers will acquire a percentage of blocks above a critical threshold for an “affordable” video playback. .

Several P2P streaming systems have been recently proposed. Their common denominator is that they try to cope with the environment described above by either proposing content diffusion graph (overlay) or a distributed scheduler for the distribution of blocks through an existing overlay.

A recent and relatively effective approach has been proposed in [9] where blocks are assigned to stripes. In contrast to the inflexible architectures where each block is distributed through a predefined tree topology, there

isn't a predefined path for the diffusion of each stripe and each stripe could be exchanged between any two nodes according to the sender's uploading bandwidth capabilities. In this system a mesh overlay is constructed wherein nodes have equal number of neighbors. Stripe reassignment, due to node arrivals, departures or bandwidth changes, takes place with the help of a dynamic decision function. Authors demonstrate in their original work that under static conditions, limited heterogeneity and sufficient uploading bandwidth capabilities among peers, their system achieves a quit high degree of bandwidth utilization and low setup time. On the other hand, the randomness of the formed mesh leads to low bandwidth utilization in highly heterogeneous environments. Finally, the cases where aggregate uploading bandwidth is equal or less than the bandwidth that is required are not evaluated.

Gradually, the focus of the research was concentrated to proposing schedulers for the efficient dynamic distribution of blocks among peers. More specifically, a new scheduling algorithm is proposed in [1] according to which, neighbors in the overlay periodically exchange buffers that contain the sets of blocks that they have received. Each time a peer (sender) has blocks to transmit, it selects the most deprived neighbor, namely, the neighbor (receiver) that misses the largest number of blocks among those blocks currently held by the sender and transmits one of its missing blocks randomly. Such a scheduler copes more efficiently with the heterogeneous uploading capabilities than in [6], it increases the levels of fairness while system has better performance during peer departures. On the other hand, duplicate block transmissions occur resulting in wasted bandwidth while any attempt to minimize those leads to the increase of setup time.

The scheduler proposed in [11] constitutes an improvement of the previous one [1]. According to it, the sender still selects the most deprived neighbor but this time it sends the most recently generated block by the source instead of transmitting a random one. By selecting the most recent block we improve the diffusion speed and we reduce the setup time. In order to deal with duplicate block transmission a negotiation mechanism among peers has been introduced and it increases bandwidth utilization. Finally, they propose an algorithm to reorganize their random mesh overlay aiming at alleviating bandwidth bottlenecks in the presence of highly heterogeneous peer capabilities. Despite these improvements their system still suffers from moderate values of setup time especially when compared with values reported in [12] and [9]. This is attributed to the non-optimized overlay and the high scheduling mechanism overhead.

In [12] the scheduler of [11] is slightly improved by prioritizing block transmissions in nodes with high uploading bandwidth. Through the analysis we observe lower setup time values that range with the distance of peers with high uploading capabilities in the overlay from the source on contrast authors do not exploit this finding through overlay construction. Furthermore we observe very low bandwidth utilization and fairness.

Based on the brief analysis above, it is evident that there is a trade off among all these factors that affect the overall performance of a P2P system. Addressing this trade off calls for a *holistic* methodology in building P2P systems for live streaming, according to which both the architecture of the overlay and the scheduler must be co-designed and optimized.

This is the scope of this paper. On one hand, it proposes an overlay architecture that clusters the nodes with high uploading bandwidth capabilities and maintains a *graph topology* whereby highly heterogeneous peers are able to contribute all of their uploading bandwidth. The overlay is continuously and efficiently reorganized by means of innovative distributed algorithms under dynamic network latencies and node arrivals and departures. To the best of our knowledge no previous work has proposed such an overlay. The overlay is complemented with a scheduler that benefits from the properties of the overlay and optimizes the way that neighbors and blocks are selected for transmission. The former are selected according to a *multi-objective sender driven neighbor selection* algorithm while the latter are selected by combining a proactive *receiver driven block selection technique and a block transmission mechanism.*

2. Overlay architecture

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.

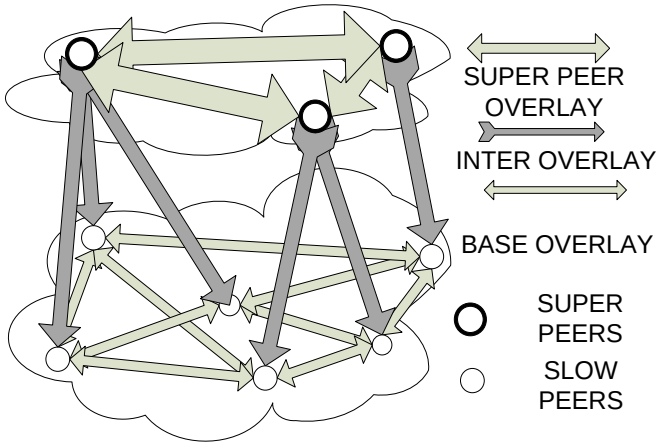


FIGURE 1. *The structure of the overlay grph.*

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 $M_I=1$.

The introduction of the super-peer overlay has been proposed with a number of objectives in mind. By clustering together peers that their uploading bandwidth is higher than the video stream rate, we guarantee fast distribution of each block among the super peers. Furthermore, by organizing the peers in a way that everyone has equal number of neighbors with the smallest possible distance (locality) ensures uniform graph connectivity and short graph diameter. This results in a graph structure that can be exploited by a scheduler to achieve fast and optimal diffusion of the stream to the super peers.

The same graph organization mechanism is also followed in the base overlay in order to have the same benefits with the super-peer overlay. Also the purpose of the *base overlay* is the utilization of the uploading bandwidth of slow peers as it is the only overlay in which they transmit blocks. The need for *inter-connections* between super peers and slow peers, and the presence of super peers in the base overlay takes place in order to utilize the excess bandwidth of super peers that it is not used in the super peer overlay. This process provides the sufficient incoming bandwidth for the slow peers in order to acquire the whole stream in the given time constrains while optimally it utilizes the uploading bandwidth of the super peers during the operation of the system.

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 in order to keep our streaming system unaffected from dynamic peer behavior.

In every real p2p live streaming system peers enter and leave the system dynamically. Due to this peer behavior it is infeasible to have always an overlay with the exact structure that we propose. On the other hand the proposed graph structure, as we present in the evaluation section, exploits the capabilities of the participating peers and enhances the performance and the stability of our system. Our objective is to continuously and dynamically reorganize the graph neighborhoods in order to have always a graph close to the proposed structure even during peer arrivals and departures.

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.

2.1. Intra-overlay distributed optimization algorithm

Intuitively the goal of this algorithm is to maintain a balanced overlay where nodes have equal number of neighbors while each node has neighbors that are physically close to it in the underlying network. This algorithm is responsible for handling: a) the arrival or departure of a node, b) changes in network conditions, and c) changes in the neighborhood set of a peer.

Intra-DOA makes use of a function, called *energy function* and 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 (single trip time), $Stt(i,j)$, of node i with all of its neighbors $j \in N(i)$, that is,

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

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

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

The set S includes all the nodes that participate in the overlay every time instant. Our algorithm is an iterative distributed algorithm that each node executes periodically and uses integer linear optimization to minimize the sum of the energies of a small fraction of nodes that participate in an iteration of it while simultaneously it balances the neighbors of the participating peers. We later prove its global convergence.

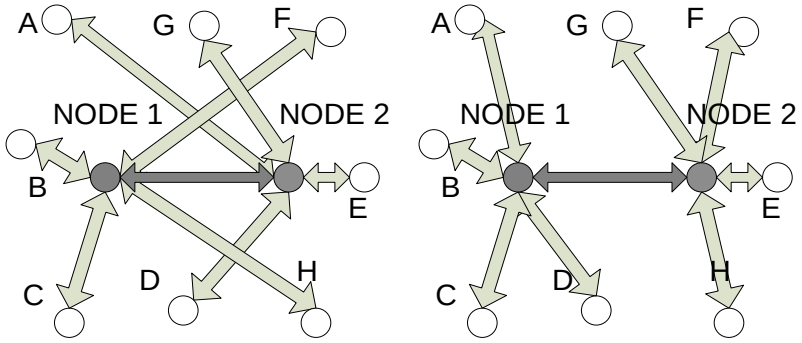


FIGURE 2. Overlay organization before and after an execution of Intra-DOA

Every time the algorithm is executed two adjacent peers in the overlay, which we call initiators, mutually exchange the sets with the network addresses of their neighbors. Then each initiator node measures the network latencies between itself and the neighbors of the other initiator node, which for clarity we call them satellite nodes. Left part of Figure 2 shows the initiators, Node 1 and Node 2, and the set of their neighbors $N_{before}(1)=\{b,c,f,e\}$ and $N_{before}(2)=\{g,d,a,h\}$ respectively. In the figure the length of the edges between two nodes is proportional to the network latency between them.

We note as $Swap$ the set of the initiators and as $N_{before}(i)$ the initial sets of satellite peers of each initiator i that could be reassigned during the execution of the Intra-DOA. We note as $\alpha(i,j)$ a parameter that if its value is equal to 1 the initiator i will become neighbor with satellite j and if become 0 it won't. The function that we want to minimize during the iteration of Intra-DOA is:

$$\mathbf{min} \sum_{\mathbf{i \in Swap}}^{\mathbf{|Swap|}} \mathbf{E(i,N(i))} = \mathbf{min} \sum_{\mathbf{i \in Swap}}^{\mathbf{|Swap|}} \sum_{\mathbf{j \in N(i)}}^{\mathbf{|N_{before}(i)|}} \mathbf{a(i,j) * Stt(i,j)} \quad (3)$$

In order to ensure the balanced properties of the overlay we introduce two types of constraints. The first constrain ensures that satellite nodes will be assigned uniformly to each initiator. For both initiators we model this constraint with the following equation:

$$\sum_{j\in\{N(1)\cup N(2)\}} a(i,j) = \frac{1}{2} * \sum_{k\in Swap} |N_{before}(k)| \quad \forall i \in Swap \tag{4}$$

The second constrain ensures that after every iteration of Intra-DOA the number of neighbors of a satellite node j that belongs to the various $N(i) \mid i \in Swap$ remains constant and is expressed by the following equation.

$$\sum_{i\in Swap}^{|Swap|} a(i,j) = |C(j)| \quad \forall j \in \{N(1) \cup N(2)\} \tag{5}$$

where $C(j)$ denotes the set of nodes that belong to *Swap* and are neighbors with *j* before the execution of the algorithm. In most cases each satellite node is neighbor with one of them.

Right part of Figure 2 illustrates the new sets of initiator's neighbors $N_{after}(1)=\{a,b,c,h\}$ and $N_{after}(2)=\{g,d,e,f\}$ respectively, after a single iteration of Intra-DOA.

Obviously the total energy (Eq. 2) of the system is a positive number as it is the sum of network latencies between nodes. If we prove that after each execution of Intra-DOA the total energy of the overlay decreases, then we have proven that our algorithm globally converges and there are no fluctuations in the overlay. By taking into consideration an execution of Intra-DOA we can rewrite the total energy of the overlay as:

$$E_{all} = \sum_{i\in S-Swap-Satellite}^{|S-Swap-Satellite|} E(i,N(i)) + \sum_{i\in Satellite}^{|Satellite|} E(i,N(i)) + \sum_{i\in Swap}^{|Swap|} E(i,N(i)) \tag{6}$$

The first term of (Eq. 6) expresses the energy of the nodes that do not participate in an iteration of the algorithm. The second term is the energy of the satellite nodes and the third term is the energy of the swap initiators. During each execution of Intra-DOA the first term of the equation remains unchanged while the third term decreases as this is the objective of the algorithm. We prove that the second term also decreases in an iteration of Intra-DOA equally to the reduction of the third term. This is due to the symmetry of the overlay since every edge participates in the energy of an initiator and a satellite node. More formally we can rewrite the energy of each satellite node as:

$$E(I,N(i)) = E(I,\{N(i)-m \mid m\in Swap\}) + Stt(i,m) = Esat(i, Nsat(i)) + Stt(i,m) \tag{7}$$

Where $Esat(i, Nsat(i)) = E(I,\{N(i)-m \mid m\in Swap\})$ is the energy of a node that remains constant during the algorithm execution.

The sum of all the energies of the satellite nodes is:

$$\sum_{i\in Satellite}^{|Satellite|} E\big(i,N(i)\big) = \sum_{i\in Satellite}^{|Satellite|} Esat\big(i,Nsat(i)\big) + \sum_{i\in Satellite} \sum_{m\in Swap} a(i,m) * Stt(i,m) \tag{8}$$

$$= \sum_{i\in Satellite}^{|Satellite|} Esat\big(i,Nsat(i)\big) + \sum_{i\in Swap}^{|Swap|} E(i,N(i)) \tag{8}$$

If we replace this term in Eq. 6 we end up with the following expression:

$$E_{all} = \sum_{i\in S-Swap-Satellite}^{|S-Swap-Satellite|} E(i,N(i)) + \sum_{i\in Satellite}^{|Satellite|} Esat(i,Nsat(i)) + 2 * \sum_{i\in Swap}^{|Swap|} E(i,N(i)) \tag{9}$$

The first two terms of this equation do not change and the third one is the minimization function that reduces the energy in every execution of Intra-DOA. This proves that our algorithm converges to a minimum energy overlay. Our system evaluation confirms the convergence of our algorithm and the vast reduction of the energy of each node.

2.2. Inter overlay distributed optimization algorithm

This distributed algorithm aims at optimizing the inter-connections that super peers use in order to assist slow peers to distribute all the blocks. It ensures that every super peer has as neighbors peers that are physically close to it in the underlying network, while the number of these neighbors is kept proportional to the excess uploading bandwidth of each super peer.

The excess bandwidth of each super peer *i* is defined as $BE(i)=c(i) - \mu$, where μ is the streaming service rate and $c(i)$ is the uploading bandwidth of i.

At the beginning of Inter-DOA two adjacent initiator peers in the super peer overlay (indicated again as Node 1 and Node 2 in Figure 3) exchange the sets $M_I(i)$ of the network addresses of their neighbors in the base overlay. For instance, assuming that Node 1 has twice as much excess bandwidth than Node 2 in Figure 3, their corresponding sets are $M_{I\_before}(1)=\{a,c,f\}$ and $M_{I\_before}(2)=\{b,d,e\}$ (left part of figure).

Using the same definitions for $Swap$ and $\alpha(i,j)$ the function we want to minimize this time is:

$$\mathbf{min}\sum_{\mathbf{i}\in\mathbf{Swap}}^{|\mathbf{Swap}|}\mathbf{E}(\mathbf{i},\mathbf{N}(\mathbf{i}))=\mathbf{min}\sum_{\mathbf{i}\in\mathbf{Swap}}^{|\mathbf{Swap}|}\sum_{\mathbf{j}\in\mathbf{N}(\mathbf{i})}^{|\mathbf{M_{I\_before}(i)}|}\mathbf{a}(\mathbf{i},\mathbf{j})*\mathbf{Stt}(\mathbf{i},\mathbf{j}) \quad (10)$$

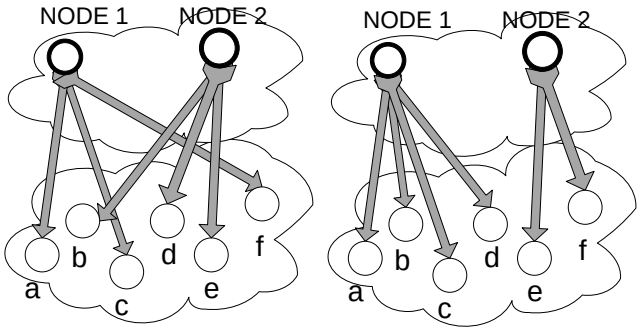


FIGURE 3. Inter overlay connections before and after Inter-DOA execution.

Inter-DOA is executed in exactly the same way as Intra-DOA and we introduce two types of constraints. The first ensures that after every iteration of Inter-DOA the number of neighbors of a satellite node j that belongs to the various $M_I(i) \mid i\in Swap$ remains constant and is expressed as:

$$\sum_{i\in Swap}^{|Swap|}a(i,j)=|C(j)| \quad \forall j\in\{Ms(Node\ 1)\cup Ms(Node\ 2)\} \quad (11)$$

where $C(j)$ denotes the set of nodes that belong to $Swap$ and are neighbors with *j* before the execution of the algorithm.

The second constraint (one for each initiator i) ensures that the participating super nodes will distribute their interconnection edges proportionally to their excess bandwidth $BE(i)$, and is expressed by,

$$\sum_{j\in\{N(1)\cup N(2)\}} a(i,j) = [|N(1)| + |N(2)|] \frac{BE(i)}{\sum_{k\in Swap} BE(k)} \quad (12)$$

Right part of Figure 3 illustrates the new sets of interconnections after an iteration of Inter-DOA $M_{I\_after}(1)=\{a,b,c,d\}$ and $M_{I\_after}(2)=\{e,f\}$.

2.3 Inter-ISP traffic minimization

We can extend our DOA in order to take also into account the ISP in which each peer belongs. In order to exploit this information and minimize the traffic between ISPs we follow exactly the same method that we describe the previous sections but this time we calculate the energy of each peer as:

$$E(i,N(i)) = \sum_{j\in N(i)}^{|N(i)|} [Stt(i,j) + p(i.j)] \quad (13)$$

Where $p(i,j)$ is zero where i and j belong to the same ISP and a large number if they belong to different ISPs. An extension of this work could be the factorization of the costs that different inter ISP links have through p we leave this now as future work.

3. Scheduler architecture

Without loss of generality, in a P2P live streaming system a source generates a video/audio stream with a service rate of μ *bits/sec* that is the object playback rate. Every second of the stream is then divided into N_b blocks. So each block is generated every $1/N_b$ *seconds* with a size equal to $L_b=\mu/N_b$ $bits$. These blocks are sent directly to a small subset of peers. Consequently, peers that are neighbours in the overlay mutually and dynamically exchange blocks in order to 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. The scheduler that runs in every peer decides which block should be transmitted next, to which neighboring peer.

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 from each one of them. In this way, we can achieve fast and complete diffusion of each block while we avoid the transmission of the same block from different sending peers to the same receiving peer. We consider this receiver driven block selection approach as the most efficient one in distributing blocks since the receiving peer always has a better knowledge about the rarity of its missing blocks in the buffers of its 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 the 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 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 in this time interval. 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 from their buffers. We note the period in which these two algorithms are executed, `request_interval`. The third component is the *neighbour selection algorithm*, which is executed by the sending peer exactly before the transmission of a block taking into account the requests from receiving peers, their upload bandwidth capabilities and the amount of blocks that they miss.

3.1 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:

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

Super peers that participate in the super peer overlay have their available uploading capabilities greater that the video serving rate, i.e. $c(i) > s$. Therefore, the rate of the generated tokens per peer is $s * request\_interval / L_b$ that is equal to $N_b * request\_interval$ which is enough for the distribution of the stream in the super peer overlay. Similarly, the number of tokens that each super peer issues to the base overlay is proportional to its excess bandwidth (BE) as defined in Section 2 and equal to $BE(i) * request\_interval / L_b$.

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

A key requirement 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 super peer i are assigned as $P(i,j) = 1/M_S$ for each j that is also a super peer, and $P(i,j) = 1/M_I$ for each j that is a slow peer. Finally, the probabilities for each slow peer i are $P(i,j) = 1/M_B$.

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 will be 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.

3.2 Proactive Block Assignment and Request

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

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

Each peer i acting as a potential block receiver from its neighbors performs the matching algorithm with a period equal to request_interval. The output of the algorithm is propagated immediately to its neighbors. Each neighbor j of peer i has a set of blocks that we note as S_j we also define a parameter c(j,k) that is 1 if peer j has block k and 0 if peer j miss block k. We additionally note the all the neighbors that had transmitted a token during the last *request_interval* of a peer i as *Neigh(i)* (all the other neighbors are excluded from the matching algorithm) and we define a parameter *a(i,j,k)* that represents the output of the matching algorithm. We allow this parameter to take two values 0 and 1 and in the second case peer i requests from j the block k. The algorithm solves a linear optimization problem that is:

$$max \sum_{j \in Neigh(i)} \sum_{k \in Sj, k \notin Si} a(i,j,k) * c(j,k) \quad (14)$$

Under the constraint to for each block k:

$$\sum_{j \in Neigh(i)} a(i,j,k) \leq 1 \quad (15)$$

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

$$\sum_{k \in Sj} a(i,j,k) \leq 1 \quad (16)$$

Intuitively the maximization function ensures that each peer will request as many blocks as possible that are available in the buffers of its neighbours. We can observe that this function seeds up the diffusion of rare or newly produced blocks than randomized request. Furthermore it maximizes the total number of blocks that will be requested and in this way we minimize the probability to have idle upload bandwidths in *Neigh(i).* For each block we introduce a constraint that prevents the request of the same block to two potential senders and in

this way we almost eliminate duplicate block transmissions. Finally, we limit the number of requests at each neighbour to be equal to, or less than one in order to allow him distribute his upload bandwidth capabilities also to other potential receiver peers.

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

3.3 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 towards nodes of high uploading bandwidths in order to achieve fast stream distribution.

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

$$d(i,j) = \frac{difference(i,j)}{per * buf\_size} - \frac{rank(i,j)}{|neigbors(i)|} \quad (17)$$

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 quantity difference 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 ordered in descending uploading bandwidth value. We have chosen to model the network bandwidths in this way in order to make our scheduler independent of the uploading bandwidth data set and as such suitable for every uploading bandwidth distribution. Moreover, buf_size is equal to $N_b * t_s$ and denotes the number of blocks that nodes exchange at each time instant. Finally, the parameter per is a constant representing the percentage of the buffer size. We have successfully experimented with values of parameter per that are between 5%-10% of the buffer size.

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

3.4 Impact of parameters on the Scheduler.

The size of the request interval plays a crucial role in the performance of our system and is accountable for the presence or not of duplicate block transmissions. When a node i starts the transmission of a new block to node j, it takes time equal to $STT(i,j)$ for that block to reach node j. If the next request interval for node j is during this time interval then node j may request the same block from another node leading to a duplicate

packet transmission. On the other hand, if node i starts transmitting a block to node j, after node j issued its new requests but before node i receives them, then again that could result in a duplicate block transmission.

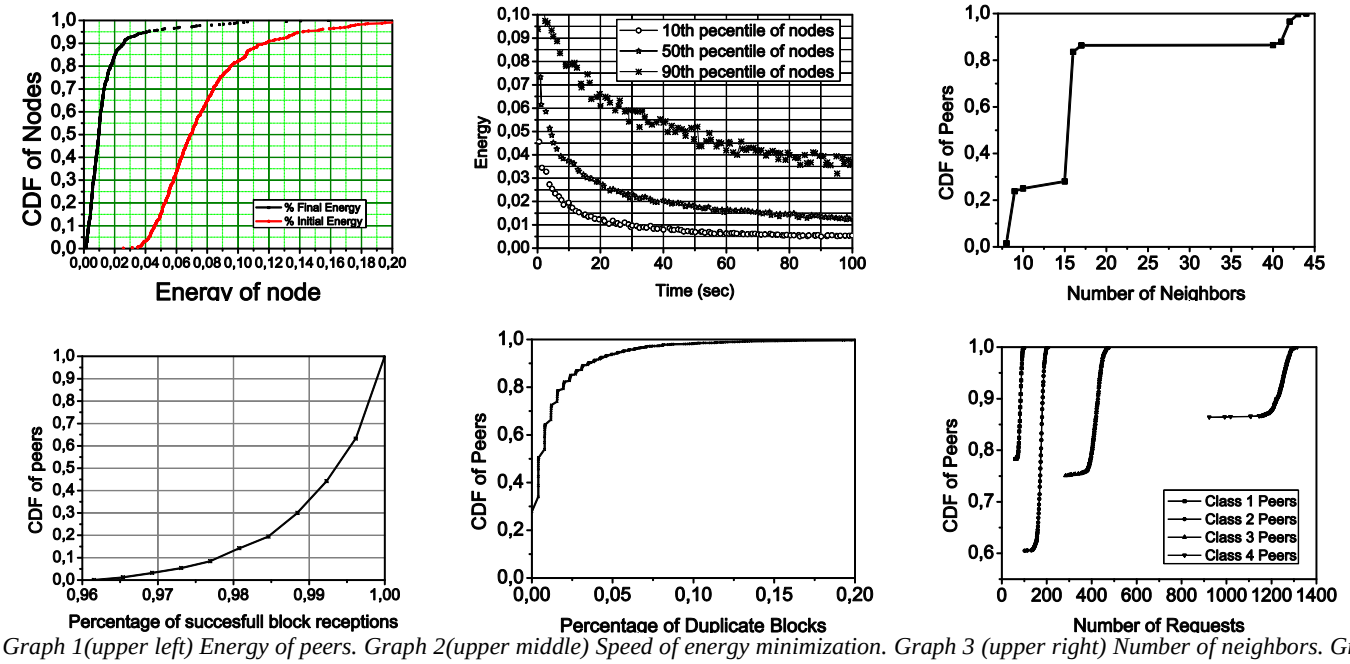
So the ratio of potential duplicate block transmissions to normal block transmissions depends on the STT values between nodes and to the size of the request interval. Large value of request interval leads to low volumes of potential duplicate block transmissions, thus increasing the maximum achievable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

4. Evaluation

For the evaluation of our P2P streaming system we have used the OPNET Modeler v.14 [8] in order to test our proposed system under various underlying network topologies and conditions. Here we present its performance based on a topology from [3] that is also used as a reference topology in [9]. However, similar performance has been observed with all topologies we have worked with. In order to model heterogeneous uploading bandwidth capabilities we set the uploading bandwidths equal to 4000 kbps (class 4), 1000 kbps (class 3), 384 kbps (class 2), and 128 kbps (class 1) corresponding to a distribution of 15%, 25%, 40%, and 20 % of peers. The average uploading bandwidth of this distribution is around 1030 kbps; this value will be used hereafter as the average uploading capacity of the system.

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

In Graph 1 we present the cumulative distribution function (CDF) 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 algorithms. 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, testifying the fast convergence properties of our optimization algorithms. Using the previous 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.



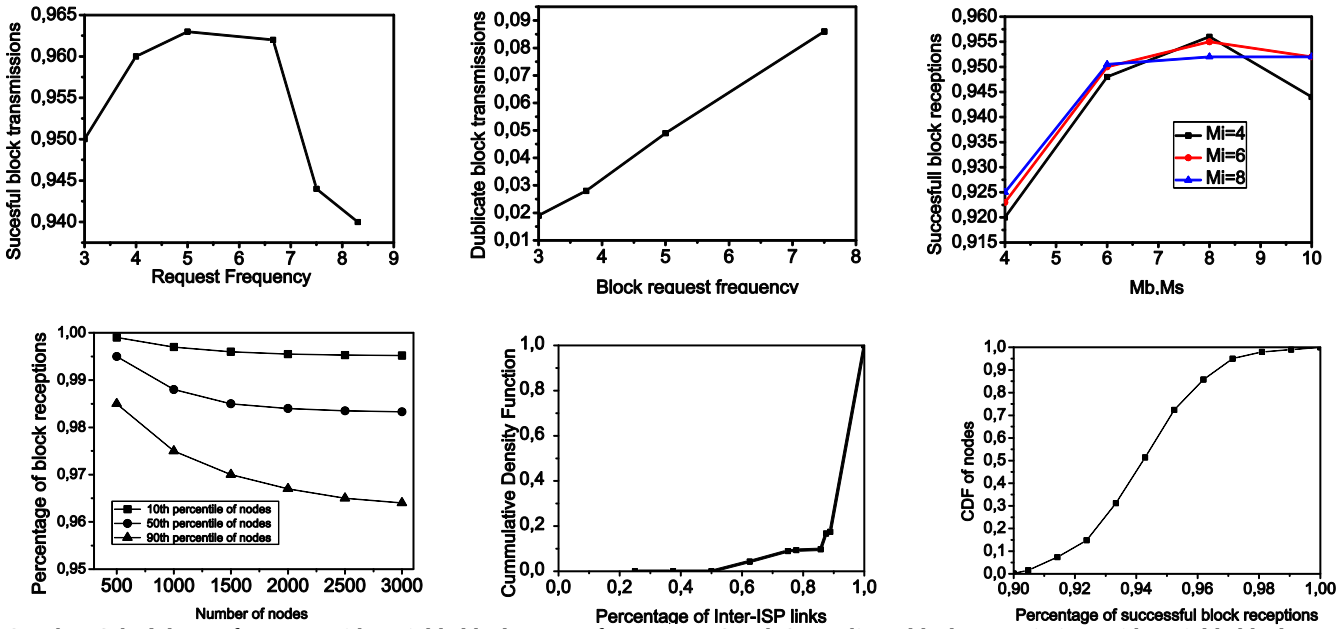
Graph 1(upper left) Energy of peers. Graph 2(upper middle) Speed of energy minimization. Graph 3 (upper right) Number of neighbors. Graph 4(lower left) Percentage of the successful block receptions of nodes in static conditions. Graph 5(lower middle) Duplicate block transmissions. Graph 6 (lower right) Number of requests for each Class of peers.

In the remaining graphs we evaluate the performance of our scheduler under various scenarios. Firstly, in graphs 4,5 and 6 we show the performance of our system under static conditions. We simulated a system with 2000 nodes in which we applied our DOA algorithms until their convergence and then started the streaming process which takes place for 50 seconds (larger values have no effect on the results). We set the values of N_b and t_s to 14 blocks and 2 sec respectively. The video streaming playback rate μ is set to the 95% of the average capacity. Finally, as in static conditions the STT 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. Our system manages to successfully deliver a stream of a video streaming rate very close to the average uploading capacities of the peers within a very small setup time period. This exhibits the high degrees of bandwidth utilization that we can achieve.

In Graph 5 we show the CDF of the percentage of duplicate packets that was received by the peers. As we can observe our scheduler in conjunction with our locality aware overlay manages to reduce the percentage of duplicate packets to around only 1% percent. The percentage, which is translated to wasted bandwidth, along with the control overhead of our system, which has been taken into account on our simulations, give us an upper bound on the maximum achievable service rate for a given distribution of upload capacities.

Analytically, The control overhead for the overlay distributed optimization algorithms is almost negligible and less than 10 kbps per node as we have derived from our simulations. Furthermore, the token transmissions, the block requests and the buffer exchanges are all exchanged between neighbors with a control UDP packet periodically with a rate analogous to $1/N_b$. This length of the packet has always been less than 200 bits



Graph 7. Scheduler performance with variable block request frequency, , Graph 8, Duplicate block transmissions with variable block request frequency, Graph 9 Variable number of neighbors in the overlay Graph 10 Performance as System Scales Graph 11,CDF of Inter-ISP Links Graph 12 System performance in dynamic conditions

including the UDP and the IP headers. In addition, as we have calculated from Graph 3 each peer has on average 20 neighbors that exchanges control packets with them. So the control overhead that our scheduler introduces is around 40 kbps that is 4% of the given average available uploading bandwidth. So the available useful bandwidth is around 95% of the average upload capacity and is the maximum achievable service rate as our system manages to utilize in an optimum way the upload capacities of the participating nodes

Another proof of how our system manages to utilize the upload capacities is shown in Graph 6 which depicts the number of block requests that each peer received. We observe here that the number of block requests is almost identical for every class of peers and proportional to their uploading bandwidth. This shows the efficiency of the token generation and the proactive block request algorithms that manage to distribute the task of block propagation evenly to every peer according to its uploading capacity.

In the next two graphs we depict the effect of the value of the request interval in the performance of our system. In graphs 7 and 8 we demonstrate the percentage of successful block receptions and the percentage of duplicate block transmissions under variable block request frequency (*1/request_interval)* respectively. The service rate is again equal to 95% of the available uploading bandwidth, the setup time is 2 sec and N_b is 14 blocks per second. From graph 7 we can observe that there is an optimal request frequency that maximizes the percentage of successful block receptions. The explanation of this phenomenon is revealed in graph 8 where we see a linear reduction of the percentage of duplicate block transmissions as the request frequency is reduced. On the other hand very small values of the request frequency lead to a condition where a sender has no blocks to transmit and so its uploading bandwidth remains idle. As we have observed through our evaluation scenarios the optimal request frequency is sensitive to the setup time and the network latencies between participating peers. On the other hand we observe that there is an interval in the request frequency from around 4 to 7 where we have a stable behavior in the performance of the scheduler and this makes us optimistic for the stability of our system although we are not in position to calculate analytically the optimal request interval.

In graph 9 we demonstrate the percentage of successful block transmissions under variable number of neighbors in the overlay with the same parameters as in the previous scenario. In all the scenarios that we demonstrate the Ms and Mb are equal in order to be able to present the results in a two dimensional graph. We can see also plots under different values of Mi. As we observe there is again an area where Mi is between 4 and 8 and Mb, Ms are between 6 and 10 where the percentage of the successful block receptions is very high and remains almost stable. This reveals the stability of the system in dynamic peer arrival and departures where the values of these parameters change temporarily.

In Graph 10 we demonstrate the behavior of our system for different numbers of participating peers. As we observe our system's behaviour is almost independent of the number of the participating nodes indicating a potential asymptotic behaviour. This makes our system very stable and promising in terms of scalability.

In Graph 11, we enhance our DOA in order to take into consideration the overall inter-ISP traffic produced by our system. In this scenario peers prefer to have as neighbours again those that are closer to them in the underlying network but also those that belong to the same ISP. In order to show the effectiveness of our DOAs we assume an extreme scenario where 2000 peers that participate with equal probability in 40 ISPs. As we can observe from this graph the 90% of the whole overlay connections are between peers that participate to the same ISP. So only 10% of the whole traffic crosses the inter-ISP network links.

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

More specifically, 2000 nodes enter our system from 0 to 200 sec (10 peers/sec) while half of them depart during the period 150 to 250 sec (10 peers/sec). In addition, the uploading bandwidth of each peer dynamically fluctuates every 2 seconds between -20% and +20% of its nominal value according to a uniform distribution. In graph 12 we show the performance of our system. We observe the degradation of our system performance is less than 3% compared to the static scenario. This shows the high tolerance of our system to dynamic conditions and also the very fast convergence of our optimization algorithms.

In Graph 13, we show again the impact that the value of the $request\_interval$ has on the performance of our system. For the previous scenario we show the percentage of duplicate block reception for a request interval equal to $3/N_b$ and $2/N_b$ with setup time equal to 3 and 2 seconds respectively. As we expected in the latter case the percentage is greater and above 5% meaning that the system can't deliver a stream of service rate equal to 95% of the available uploading bandwidth. If we want to deliver a stream with setup time equal to 2 seconds we need to degrade the service rate to about 87% of the available bandwidth. This shows that the value of the request interval is a trade-off between the maximum achievable service rate and minimum setup time.

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

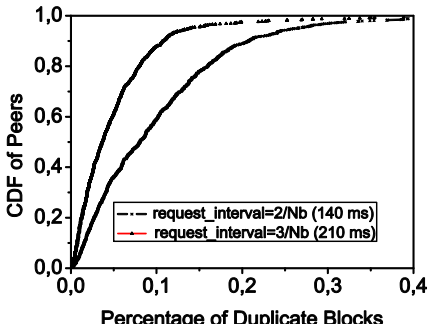
For evaluating node departures we have used an overlay with 2000 peers that it suffers from a departure of 1000 peers under various departure rates. We observe that our system is almost immune to peer departures due to the fast reconfiguration of our overlay and our dynamic neighbor selection function.

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

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

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

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



Graph 13. *Duplicate block transmissions under variable request intervals*

5. High level comparisons with other systems

As we have observed from the evaluation, in static conditions our system delivers a video stream with a setup time of 2 seconds and with a bit rate 95% of the average uploading bandwidth. This performance is better than the one reported in [6] wherein the same uploading bandwidth distribution of peers achieves a setup time of 6-20 seconds depending on the source uploading bandwidth. Furthermore, the data set used in [6] is based on emulating 80 peers while we have simulated a data set of 2000 geographically distributed nodes.

In [9] there is a much lower level of heterogeneity of uploading bandwidth distribution among peers while the authors used the same data set [3] in order to model the network latencies between peers for their experiments but with a population of 409 peers. Their architecture becomes unstable when the video playback rate exceeds 90% of the average available uploading bandwidth. Our system also achieves better setup time than the one reported which is more than 5 seconds.

Finally, [12]uses high levels of bandwidth over provisioning with a video playback rate of around 55%-60% with setup time that vary significantly between 2 and 10 seconds among the participating nodes. More detailed comparisons between our work and other systems can be found in [17].

6. Conclusions

In this paper we have proposed and evaluated a complete P2P live streaming system that adopts a holistic approach that involves co-designing of the overlay and the scheduler architecture. Our system achieves a high level of performance as it is capable of delivering the stream with a service rate very close to the average available uploading bandwidth of the participating peers, at very low set-up times even under dynamic network conditions and peer behavior. This result has been possible due to an innovative overlay architecture that takes into account locality information among peers and exploits their heterogeneous uploading capabilities. This overlay system is based on two newly proposed real-time distributed algorithms also tolerant to dynamic behavior of peers. Finally, a new scheduler, co-designed with the overlay, distributes the video stream in an optimal manner. We believe that only under such a holistic approach a successful P2P live streaming can be realized.

Acknowledgements

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

8.References

[1] Laurent Massoulie, Andy Twigg, Christos Gkantsidis, Pablo Rodriguez Randomized decentralized broadcasting algorithms. IEEE INFOCOM 2007

[2] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, Atul Singh, SplitStream: High-Bandwidth Multicast in Cooperative Environments, ACM Symposium on Operating Systems Principles, Proceedings of the nineteenth ACM symposium on Operating systems principles 2003

[3] Bernard Wong, Aleksandrs Slivkins, Emin Gün Sirer, Meridian: A Lightweight Network Location Service without Virtual Coordinates. SIGCOMM Conference, Philadelphia, Pennsylvania, August 2005

[4] A. Rowstron and P. Druschel, Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems, Proc. Middleware, 2001

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

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

[7] Dimirti P. Bertskeas, Network Optimization: Continuous and Discrete Models, Athena Scientific, May 1998

[8] www.opnet.com

[9] Meng ZHANG, Qian ZHANG, Lifeng SUN, Shiqiang YANG, Understanding the Power of Pull-Based Streaming Protocol: Can We Do Better?, IEEE JSAC 2007

[10] C. H. Ashwin R. Bharambe and V. N. Padmanabhan, Analyzing and Improving a BitTorrent Network Performance Mechanisms. IEEE INFOCOM, 2006

[11] Fabio Picconi and Laurent Massoulie, Is there a future for mesh-based live video streaming? IEEE P2P 2008

[12] Ana Couto, Emilio Leonardi, Marco Mellia, Michela Meo, A Bandwidth-Aware Scheduling Strategy for P2P-TV Systems IEEE P2P 2008

[13] N. Sklavos, K. Touliou, Power Consumption in Wireless Networks: Techniques & Optimizations, International Conference on "Computer as a Tool", IEEE EUROCON 2007

[14] 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.

[15] D. Wu, Y. Liu, K.W. Ross, Queuing Network Models for Multi-Channel Live Streaming Systems, IEEE INFOCOM 2009

[16] Dan-Cristian Tomozei, Laurent Massoulie, Flow Control for Cost-Efficient Peer-to-Peer Streaming, INFOCOM 2010

[17] Athanasios Christakidis; Nikolaos Efthymiopoulos; Spyros Denazis; Odysseas Koufopavlou, On the architecture and the design of P2P live streaming system schedulers, International conference on ultra modern telecommunications ICUMT 2009