On the architecture and the design of P2P live streaming system schedulers.

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

Abstract -- ***In this paper we analyze P2P live streaming systems. Through this analysis we obtain the crucial parameters for their performance in terms of bandwidth utilization, set-up time, fairness and stability. We propose a sender driven multi objective decision function for neighbor selection in order to adapt the distribution of available bandwidth to the overlay connections while simultaneously we further exploit the locality properties of an overlay. At last we develop and apply a receiver driven block selection with a content diffusion optimization algorithm that achieves fast and efficient diffusion of every block. The evaluation of our system reveals its very high levels of performance in terms of setup time, bandwidth utilization, its fair behavior in the distribution of available aggregate bandwidth in various nodes and its stable behavior as system grows and aggregate upload bandwidth changes. Finally by comparing our system with other recently developed we observe that it vastly outperforms.***

Keywords- p2p live streaming, distributed scheduling

I. INTRODUCTION

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

Peers involved in these systems have heterogeneous uploading bandwidth capabilities while the average uploading bandwidth capability of the participating peers constrains the maximum service rate of the video stream that can be delivered successfully to all peers[9]. Accordingly, an efficient P2P streaming system must be able to deliver a video stream with a service rate as close as possible to the average uploading capability of the participating peers with the smallest possible delay, called *setup time*. As setup time we define the time interval between the generation of a block from the origin server and its distribution to every peer in the system.

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

Several approaches that have been recently proposed for creating P2P streaming systems may fall into two major categories characterized by the way the stream is diffused in all nodes.

In the first category the diffusion of the stream is dictated by the graph of the overlay in which nodes participate. The pioneer representative of this category is Spitstream [4][10]. Spiltstream uses a formation of trees whereby each node is leaf in every node but one. These trees are derived from a locality aware DHT called Pastry. Blocks are assigned equiprobably into a number of stripes equal to the number of trees. Each tree distributes one stripe by propagating each one of its blocks from parent to its children. SplitStream and systems alike have two advantages. They are topologically aware (trees are formed according to the network distance between nodes) and the diffusion of blocks is done through predefined paths according to the graph topology. These lead to smaller setup times as the propagation of a block from the root of the tree towards the leaf nodes is done through nodes which may be physically close in the underlying network without any control overhead. However these systems suffer from two main drawbacks: a) they don't take into account the heterogeneous and changing uploading capacities of the peers, and b) they cannot react quickly enough to the dynamic behavior of the participating peers and the underlying network, as observed in commercial P2P streaming systems [18]. As a result these systems exhibit a low upload bandwidth utilization of the participating peers and they can't guarantee the stability of the video playback.

In the second category the stream is diffused with the help of a scheduler that resides in every peer. In these systems the peers are part of a mesh overlay. Each node maintains connectivity with a small subset of nodes which are considered as being its neighbors in the overlay. Blocks that are generated from the server are assigned play-back deadlines. Each peer maintains a number of lists (buffers), one per neighbor. Each one of these buffers contains missing blocks by its neighbor that their playback deadline has not expired yet. This information is exploited by schedulers running at each peer that are responsible for exchanging blocks in order to diffuse the stream to every peer.

We can distinguish two types of schedulers. In sender driven schedulers, the sender decides to which neighbor must send the next block. In [2][2], the selection criteria used by the sender concern the most deprived node, namely the node that misses the largest number of blocks. When the deprived node is found, a randomly chosen block is forwarded during the next transmission.

The main advantage of these systems is their flexibility that allows them to exploit the heterogeneity of the participating peers and deal with the dynamic behavior of the network. This leads to higher levels of bandwidth utilization by optimizing the flows among the participating nodes. However, this flexibility introduces a bandwidth overhead due to large numbers of buffers exchanges and duplicate block transmissions. The later is attributed to the fact that during the propagation of blocks in an unstructured overlay, each block may follow different paths before reaching a node resulting in duplicate receptions. Finally, the lack of locality awareness in the random mesh that is used as a graph and the additional phases of negotiations for the block exchanges, introduce large setup time values in the systems that follow this kind of architecture. Only in AnySee there is a mechanism for reflecting locality in the random mesh, by exchanging neighbors that are not so close in the underlying network with other closer nodes, at the expense, though, of a greatly unbalanced graph.

In contrast, in receiver driven schedulers, the receiver explicitly requests from a sender which block should be transmitted next. In Prime [15][15], the server of the stream constructs a spanning tree out of the mesh overlay, putting the server at the root of the tree. Then the participating peers periodically request from their parents in the tree the newly created blocks, while they request any remaining blocks from other neighbors in the mesh overlay.

The main advantage of these systems is the high levels of bandwidth utilization due to the overlay construction mechanism and the elimination of duplicate block transmissions due to the receiver driven scheduler. On the other hand they suffer from very large values of setup time, lack of fairness and temporary waste of bandwidth in case of more frequent node departures.

This paper proposes a novel architecture of a P2P live streaming scheduler. It is designed in such a way that synergistically optimize the trade-offs observed among various parameters impacting the performance of P2P streaming systems.

The P2P overlay is a *symmetric locality aware and self-organized overlay* that is analyzed in [21] where nodes are organized in such a way that they have similar number of neighbors and they can be dynamically reconfigured according to changes in the underlying network conditions.

The other component, the scheduler, has been designed to perform two separate but interdependent decisions: selecting the next neighbor and selecting the specific block to transmit to. The former is performed according to a *multi-objective sender driven neighbor selection* process based on an algorithm that optimizes the flows in the overlay according to the capacities of the nodes [2]and exploiting locality information of our overlay. The latter is performed according to a *receiver driven block selection process with a content diffusion optimization algorithm* that achieves fast and efficient diffusion of every block while considerably reducing duplicate block transmissions. These result in a P2P live streaming system that exhibits very small setup times and high levels of bandwidth utilization. Furthermore, we observe high degrees of fairness in upload bandwidth utilization among all nodes and a very stable and scalable system as it grows in numbers of participating peers.

The rest of this paper is organized as follows. In Section II we briefly present P2P live streaming systems and we describe our P2P live streaming scheduler. In Section III we evaluate its performance. Finally, in Section IV we give our conclusions and the future work.

II. P2P STREAMING SYSTEM SCHEDULING

Without loss of generality we assume that in a P2P streaming system there is a bootstrap node which is used for the admission of the nodes in the system while it acts as a source for providing the video stream. Furthermore, the video stream is divided into blocks. The block size depends on the service rate, say μ (measured in bps that the video playback requires), and the number of blocks in which the bootstrap node divides one second of video playback. We define this number as N_b *blocks/sec* representing also the frequency of new blocks generated by the source. So each block is generated every $1/N_b$ *seconds* at the bootstrap node, with a size equal to $L_b=\mu/N_b$ *bits*.

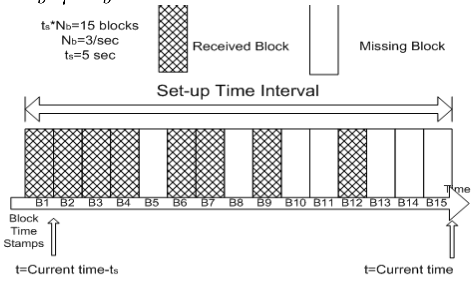


Figure 1. Snapshot of a buffer in a node with the states of the blocks.

Every block is also associated with a time stamp indicating the time of its generation. All peers reproduce (play) the video with a delay called *set-up* time which we denote it as t_s. As mentioned previously, setup time is the time that elapses from the generation of a block at the source until its distribution (propagation) to every node in the P2P system. Accordingly, at every time instant every peer plays the block that was generated t_s times before in the origin server, provided of course that this block has eventually reached its destination.

During this setup time a number of blocks have been generated, equal to N_b*t_s, the first of which will be played by every node after t_s seconds. Therefore, at every instant every node is required to keep track of all N_b*t_s blocks generated within a sliding window of t_s seconds. For this reason every node maintains a buffer of size N_b*t_s that holds the state of these blocks. Two states are of interest: received blocks and missing blocks (not delivered yet). Figure 1 provides a snapshot of the states of blocks of a buffer in a node.

Whenever the origin server produces a new block it forwards it first to a small subset of the peers (even one) which participate in the system. The effect of the size of this subset in the performance of the system is analyzed in [18][18]. Each peer maintains connections and exchanges blocks with a relatively small number of nodes, which we call its neighbors, in order to retrieve the whole video stream. To know exactly which blocks should be exchanged, each peer exchanges the contents of its buffer with every one of its neighbors. Then a scheduler that runs in the nodes decides which block should be transmitted next to which neighboring node.

A. *Scheduler for neighbour selection.*

Our scheduler extends the selection criteria and vastly improves its performance by also taking advantage of the network distance between neighbours provided by the locality aware overlay. Starting from sender driven schedulers like the one described in [2], our proposed scheduler is enhanced with a decision mechanism that takes into account short propagation paths based on STT values. As STT (single trip time) we express the network latency between two nodes that are neighbours in the overlay. We measure this latency dynamically for the dynamic overlay optimization and we can exploit this knowledge of the conditions in the underlying network in our scheduler.

In order to achieve this goal we define a decision function, d(i,j) that provides a metric used for the selection of a neighbouring node j for block transmission by node i. The decision function is given by the following formula:

$$d(i,j) = \frac{diference(i,j)}{per * buf\_size} - \frac{rank(i,j)}{|neigbors(i)|} \qquad (1)$$

The node selected for block transmission is the one with the maximum d(i,j) ∀j. In this equation |neighbours(i)| denotes the total number of neighbours of i. Additionally `rank(i,j)` is a function that returns the position of node j in a list of nodes ordered in incremental STT value with node i. We have chosen to factorize the network latencies between i and its neighbours in this way in order to make our scheduler independent of the STT values and as such suitable for every underlying network topology. Finally, `buf_size` is N_b*t_s as described in section II and denotes the number of blocks that nodes exchange at each time instant. Finally, parameter `per` is a constant representing the percentage of the buffer size. We have successfully experimented with values of parameter `per` close to very small percentages of buffer size (between 5%-10%).

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], with 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 while the diffusion of blocks is done according to network locality in order to achieve fast block propagation. As we have observed through our simulations this is taking place in most of the cases. On the other hand, when differences for missing blocks in the order of `per*buf_size` are observed, our locality aware scheduler approximates the previous 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.

B. *Scheduler for block selection*

In the previous section we have analyzed the factors that affect the selection of a node for block transmission. In this section we will focus on the mechanism that determines which block should be sent to the selected node aiming at minimizing duplicate block transmissions and fastly diffusing newly produced or rare blocks within an overlay “neighborhood”. The decision is receiver driven, in other words the sender has been notified by the potential receiver about the block the receiver wishes to receive.

Due to the symmetric property of the locality aware overlay, a receiver node is already informed about the buffer contents of its neighbors. Therefore, by applying a matching process a receiver node can proactively request different blocks from its neighbors thus resulting in the reduction of duplicates. The matching process is accomplished by performing a weighted matching algorithm between the missing blocks and those neighbors that have them, while favoring those nodes that are closer to the requesting node since they have higher chances for selection. Accordingly, whenever a node’s scheduler chooses the next node for transmission, it also takes into account whether the selected node has already requested a specific block, otherwise it selects one randomly.

Ideally, our algorithm could have accomplished complete elimination of duplicate block transmissions. However, it takes time equal to STT(i,j) for a request message to reach node j from node i. During this time, node j may have already transmitted a block to node i other than the requested block. As the blocks that node j can send to node i, are STT(i,j)/t_{send}(j) the efficiency of our algorithm depends on two parameters: a) the small values of STTs, which are provided by our locality overlay, and b) the N_b as it is the parameter that influences the time t_{send} (eq. (4)). The specific values of these parameters eventually put an upper bound on the performance of our algorithms as it is shown in our evaluation tests.

III. EVALUATION

For the evaluation of our P2P streaming system we have used OPNET Modeler [23] in order to avoid the imperfections of a custom made simulator. We have tested our proposed system under various underlying network topologies topology from [5], where the provided round trip time measurements were gathered using the King method between globally distributed DNS servers. In all topologies we have observe similar behavior of our system. We have opted for this particular real data set in order a) to avoid inaccurate conclusions which a network model may introduce, and b) to use a real topology of globally distributed nodes and so have a fair benchmark for a locality aware overlay without concentrations of peers in specific regions that favor our system.

First in order to present our evaluation with an accurate way we present the input parameters that affect the behavior of a P2P live streaming system, the trade-off of which needs to be efficiently optimized and we give a short definition of them.

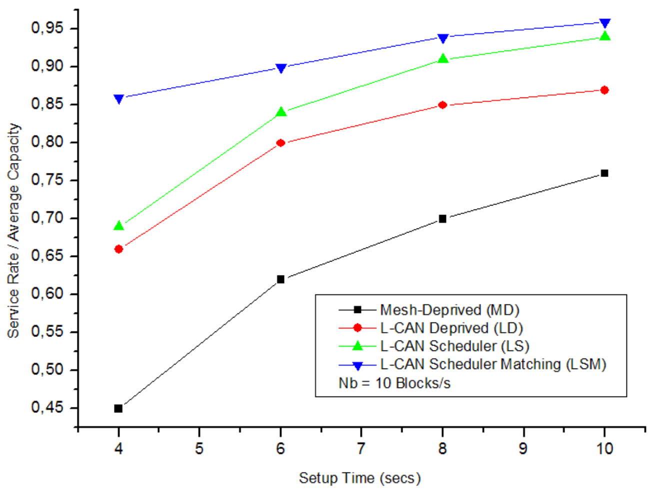
Parameter	**Definition**
Number of neighbors	The set of nodes that a node uses in order to exchange blocks.
Service rate	The playback byte rate of the video.
N_b	The number of blocks that a second of video is divided
Number of nodes	The number of participating nodes in the system a given time instant.
Average capacity	The aggregate upload bandwidth that participating nodes have divided by their number
Duplicate blocks	Blocks that transmitted to a specific node from multiple senders

Table 1. Major input parameters that affect the functionality of a p2p live streaming system.

Evaluation criteria	**Definition**
Setup time	The time interval between the generation of a block from the origin server and its distribution to every peer in the system
Maximum achievable service rate	The maximum playback byte rate of the video that a system is able to deliver to every participating node
Fairness	The equal utilization of the upload bandwidth among receivers in order to have a percentage of blocks successfully delivered to *every* node

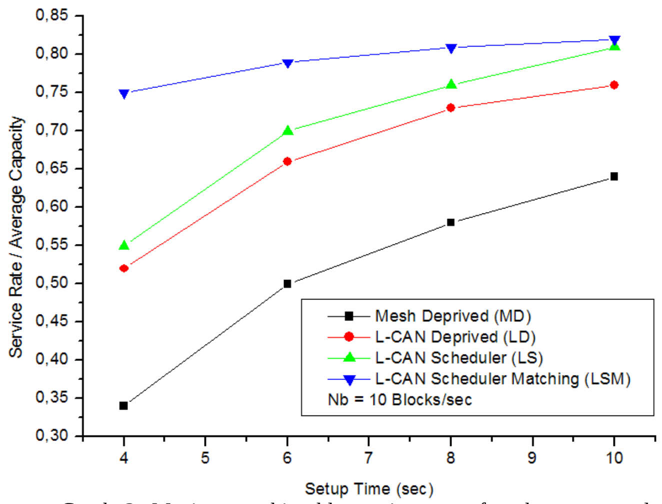
Table 2. Major evaluation criteria that we present

Graphs 1 and 2 show the ratio between the maximum achievable service rate that each system can deliver and the average capacity, for various setup time values given a constant rate for block generation (Nb=10 blocks/sec). As will see in the rest of this section by selecting a different a value of Nb our system has better performance but in this point we want to present the general trends and behavior of our system. For the Graph 1 we have used homogenous upload capacities whereas Graph 2 is based on heterogeneous. Inspecting the two graphs, we observe that the same performance trend emerges from either case i.e. homogeneous and heterogeneous upload capacities. Furthermore, the same system performs slightly worst in case of heterogeneous upload capacities. This is because in the case of a highly heterogeneous environment, although our system has much better performance than recently proposed, a small percentage of upload bandwidth is wasted due to heterogeneity. That problem can be solved by the creation of virtual nodes in L-CAN [21] that will have approximately equal upload bandwidth. We leave this architecture as future work.



Graph 1. Maximum achievable service rate divided by average upload bandwidth of each system under various setup time intervals. Nodes in each system contribute equal upload bandwidths

Applying a locality aware overlay, L-CAN, results in a significant increase of the achievable service rate (LD), as opposed to a mesh overlay (MD), because of the smaller STT values that exist between the neighbours in L-CAN. Moreover, introducing a scheduler that further exploits locality using our L-CAN, the LS system further increases the service rate, especially when the size of the peer's buffers increases (buffer_size=setup-time/Nb). This is due to the fact that small sizes of the node's buffers result in a larger probability for duplicate packets which mitigates the benefits of our scheduler.



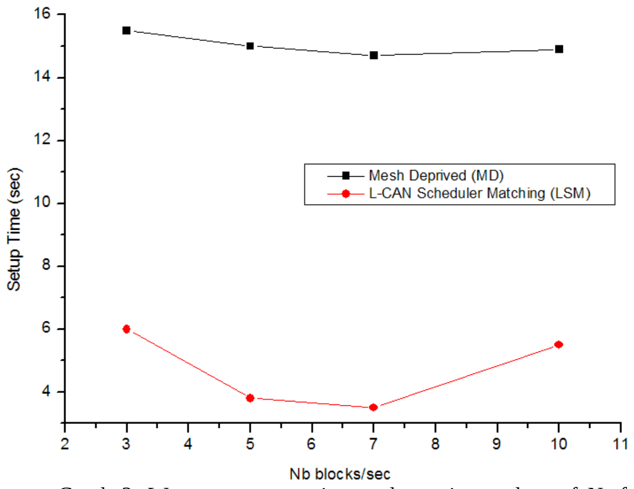
Graph 2. Maximum achievable service rate of each system under various setup time intervals. Nodes in each system contribute heterogeneous upload

Further enhancing LS system with our matching algorithm, the LSM system can achieve a maximum service rate equal to 85% and 75% of the average capacity of the participating nodes, for the homogeneous and heterogeneous scenarios respectively, even for very small setup times (4 seconds in our example). This is due to the reduction of duplicates, where in the case of LSM, a node received 15% of duplicates, whereas a node in the LS system received 30% of duplicate blocks.

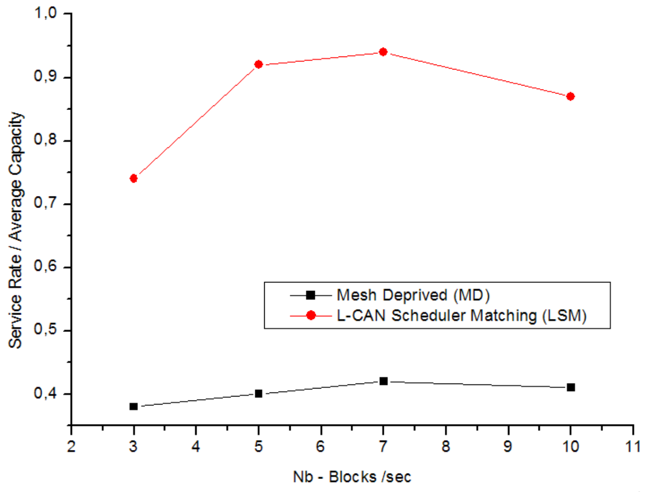
Finally, we observe that the increase of the setup time has a smaller effect in the increase of the achievable service rate in the case of LSM when compared with the other three systems. This is due to the fact, that our matching algorithm can't reduce the percentage of the duplicates packets below certain asymptotic thresholds (approximately 5% and 18% in the homogeneous and heterogeneous scenarios, respectively) as the large value of N_b=10 results in reaching the upper bound of our matching algorithm's performance as discussed in section III.E.

The Graph 3 and 4 illustrate the trade-off between setup time and N_b which in fact defines an optimum for the operation of a P2P streaming system as we also predicted in section II. Our proposed system captures this optimum. In Graph 3, assuming an application which requires a service rate equal to 90% of the average upload bandwidth, we depict the setup time that MD and LSM needs in order to deliver that service rate. We present this setup time as a function of N_b. We observe that the optimum is achieved at the point N_b=7 with a setup time 3.4 secs.

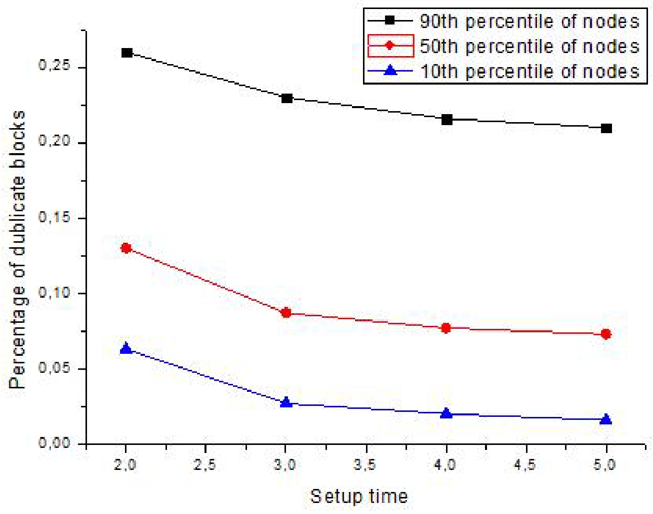
Alternatively, in Graph 4 we assume a delay intensive application, which has a setup time requirement equal to 4 seconds, and we present under different values of N_b the maximum achievable service rate again for MD and LSM. In this case the optimum is achieved for N_b=7 at the 96% of the service rate.



Graph 3. We present setup time under various values of N_b for a service rate equal to the 90% of the average capacity.



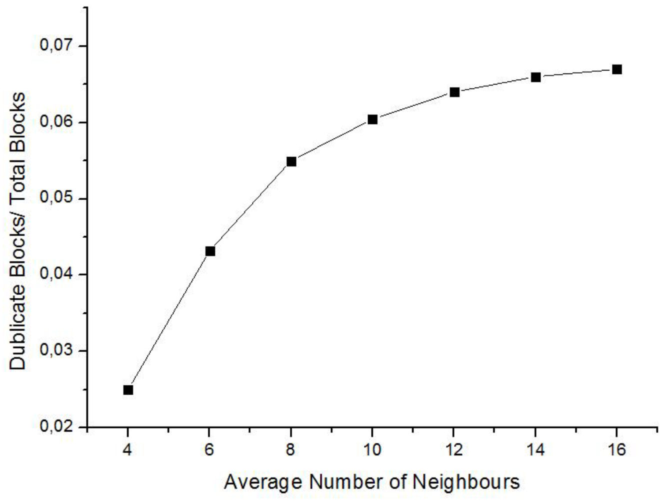
Graph 4. Maximum achievable service rate under various values of N_b for a setup time equal to 4 seconds



Graph 5. Percentage of the duplicate block transmissions as a function of the setup time(LSM).

Through Graph 5 we can observe that for values of setup time ranging from 3-5 sec we still achieve a small percentage of duplicate block transmissions with a mean value around 7%-8%. On the other hand for setup times below 3 seconds we have started to observe higher percentages of duplicate block transmissions that vastly increase below 2 seconds. As should be evident in the case of set up time 2 seconds, our system won't be able to deliver the whole stream, which needs 90% of average capacity, as a mean of 15% of the nodes upload bandwidth is wasted through duplicate block transmission. In this case we need a much more sophisticated scheduler that will exploit the probabilities with which nodes exchange blocks and make also multiple requests. We leave this as a future work.

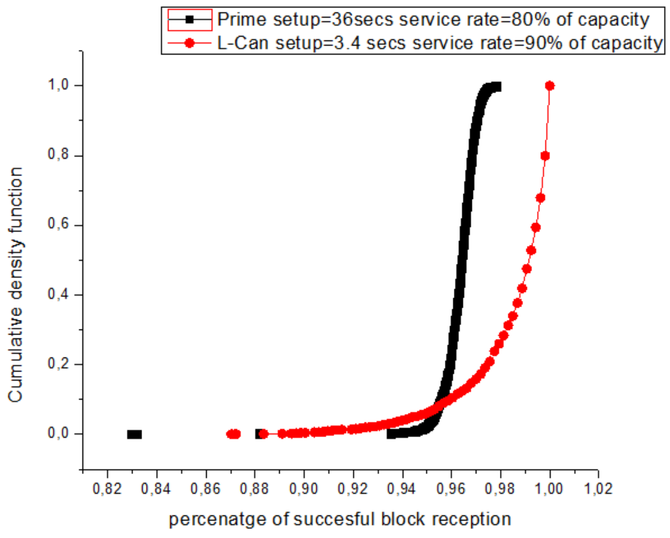
Another parameter studied in our evaluation is the impact of the number of neighbours that each node has. In graph 7 we present the percentage of duplicate block transmissions as a function of average neighbours per node (determined from L-CAN dimensions). The service rate in this case is 90% of the average capacity and again it is simulated using a system with 2000 nodes. The value of N_b is 7 blocks per second and the setup time is 3.4 seconds. As we observe duplicates are low for small values of neighbours and increase fast until they asymptotically reach 6%- 7% for neighbourhood sets larger than 12.



Graph 7. Percentage of duplicate block transmissions as a function of average number of neighbours.

Finally, in the last section of our evaluation we compare our streaming system with two others that have been already proposed.

This system, that we compare ours to, is Prime [15]. As we have already mentioned in our introduction, Prime is a system with a receiver driven scheduler. Nodes form a random mesh in which the root node, and the server of the stream, forms a spanning tree. A node in Prime distinguishes the neighbour which is its parent in the tree from its other neighbours. From its parent it requests the more recent blocks while from the others the remaining. Prime uses a parameter, called diffusion interval Δ, which defines which blocks in a node's buffer should be consider the most "recent". The other parameter is ω which multiplied by Δ defines the set up time of the system.

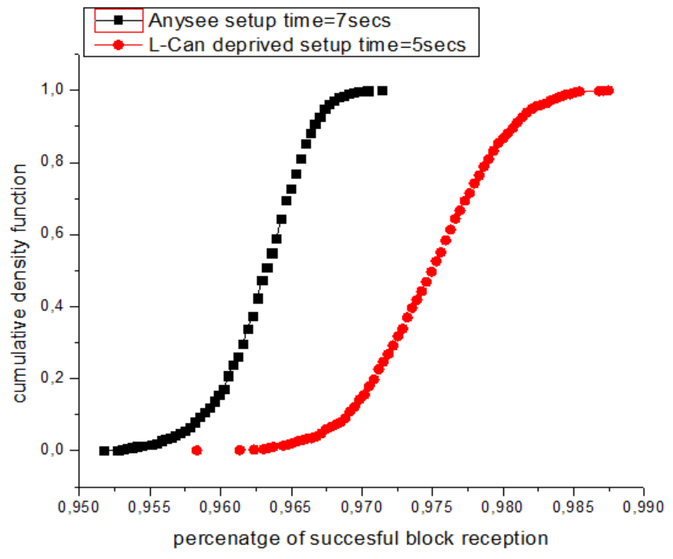


Graph 5. CDF of the percentage of successful block receptions in Prime and LSM for 2000 nodes

In order to compare it with our system we simulated a Prime system with 2000 nodes using the parameters that are used in the original Prime paper. We set Δ=6secs and ω=6 for a total setup time of 36 sec. As we didn't find any value for the Nb in the original work we run several simulations and found that the optimum value was 10. In graph 5 we compare a Prime system with the above configuration and with service rate equal to the 80% of the average capacity with our system in which we have Nb=7 blocks per second, setup time=3.4 secs and service rate equal to the 90% of the average capacity. As we can see the two systems exhibit the same behavior even if our system has around 90% smaller setup time and 12% higher service.

On top of a more balanced version of Anysee in order to improve also its performance, we applied the most deprived scheduler, as our matching scheduler requires a bidirectional graph. In order to compare only the locality properties of our system with Anysee, we also applied over our locality graph the most deprived scheduler. We simulated both the above systems with 1000 nodes, service rate equal to 90% of the average capacity and Nb=10 blocks per second. The results are presented in graph 6.

As we can observer from graph the two systems exhibits the same behavior even if our system with just the deprived scheduler has smaller setup time, which can be further, reduced with the application of our content diffusion algorithm. We highlight here that this algorithm cannot be applied in Anysee due to its asymmetric architecture.



Graph 6. CDF of the percentage of successful block receptions in Prime and LSM for 2000 nodes

IV. CONCLUSIONS AND FUTURE WORK

We have shown that a locality aware overlay increases the bandwidth utilization and reduces in a vast degree the setup time. Furthermore, we have proposed a locality aware based scheduler which also guarantees the fair block diffusion. Finally, we have developed a technique for optimized content diffusion between peers and blocks that greatly reduce the percentage of duplicate block transmissions while the control overhead remains negligible. Our design choices of our system and the algorithms thereof have been justified and motivated by the analysis of and the observations drawn from of the proposed model.

Our future work will focus on three areas of research as a direct result of our evaluation findings. The first is the creation of an architecture that is self-organized and handles the vast heterogeneity in terms of peer upload bandwidths. The second is the development of a more sophisticated scheduler that exploits exchange probabilities and makes multiple block requests. At last we are currently work on a theoretical model in order to define analytically the optimal value for stream fragmentation and the optimal number of neighbors.

REFERENCES

[1] S. Ratnasamy *et al.*, A Scalable Content Addressable Network, Proc. ACM SIGCOMM, 2001

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

[3] Ashwin Bharambe, Cormac Herley, and Venkata Padmanabhan, “Analyzing and improving bittorrent performance,” Tech. Rep. MSR-TR- 2005-03, Microsoft Research, feb 2005

[4] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, Atul Singh, SplitStream: High-Bandwidth Multicast in Cooperative Environments *SOSP’03*

[5] http://www.cs.cornell.edu/People/egs/meridian/data.php

[6] http://pdos.csail.mit.edu/P2Psim/kingdata/

[7] Castro, M., Druschel, P., Hu, Y. C., and Rowstron, A. Exploiting network proximity in distributed hash tables. In International Workshop on Future Directions in Distributed Computing (FuDiCo) June 2002

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

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

[10] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, Atul Singh, SplitStream: High-Bandwidth Multicast in Cooperative Environments *SOSP’03*

[11] “PPLive,” http://www.pplive.com.

[12] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. In *Proceedings of ACM SIGCOMM*, August 2005.

[13] Nazanin Magharei, Reza Rejaie, Yang Guo, Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches, INFOCOM , Anchorage, Alaska, 2007

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

[15] Nazanin Magharei, Reza Rejaie, PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming, INFOCOM , Anchorage, Alaska, 2007

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

[17] Peter Pietzuch, Jonathan Ledlie, and Margo Seltzer, Supporting Network Coordinates on PlanetLab, In Proceedings of WORLDS 2005

[18] Xiaojun Hei, Yong Liu Keith W. Ross, Inferring Network-Wide Quality in P2P Live Streaming Systems, Technical Report http://eeweb.poly.edu/faculty/yongliu

[19] www.opnet.com

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

[21]Nikolaos Efthymiopoulos, Athanasios Christakidis, Spyros Denazis, Odysseas Koufopavlou, “ L-CAN: Locality aware structured overlay for P2P live streaming”, in Lecture Notes in Computer Science, Volume 5274, Springer, 2008