L-CAN: Locality Aware Structured Overlay for P2P Live Streaming

Nikolaos Efthymiopoulos, Athanasios Christakidis, Spyros Denazis, and Odysseas Koufopavlou

Department of Electrical and Computer Engineering
University of Patras
Patras, Greece
{nefthymiop,schristakidis,sdena,odysseas}@ece.upatras.gr

Abstract. A p2p streaming system must be able to exploit the locality information between peers, in order to deliver a stream quickly to all peers with high level of bandwidth utilization. In this paper we propose a locality aware and balanced overlay for p2p live streaming which can adapt to the dynamic behavior of the participating peers and the underlying network. Our overlay is created and maintained through the use of two algorithms, called the placement and the swapping algorithm that we consider as the major contributions in this paper. These are responsible for the insertion of a node and the dynamic and distributed optimization of the overlay in order to reflect the underlying network. The proposed overlay is evaluated through extensive simulations that show that the bandwidth utilization of the peers and the set-up time are significantly improved through locality between peers.

Keywords: P2P live streaming, DHT, locality aware overlay.

1 Introduction

P2P streaming is a real time application with strict delivery time constraints and 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. As a final step, all peers exchange these blocks in order to reproduce the whole video stream.

Peers involved in these systems, may have heterogeneous upload bandwidth capabilities while the average upload bandwidth capability of the participating peers constrains the maximum service rate of the video stream that can be delivered successfully to all peers [12]. An efficient P2P streaming system must be able to deliver a video stream with service rate as close as possible to the average upload capability of the participating peers with the smallest possible delay, called *setup time*. With the term 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.

Several approaches that have been recently proposed for creating P2P streaming systems may fall into two categories.

G. Pavlou, T. Ahmed, and T. Dagiuklas (Eds.): MMNS 2008, LNCS 5274, pp. 77–90, 2008.

The first is based on a formation of forests of trees whereby each node is a leaf in every tree but one. Blocks are assigned equiprobably into a number of stripes equal to the number of the formed trees. Each tree distributes (pushes) one stripe by propagating each one of its blocks from parent to its children. In this category blocks are pushed according to the overlay topology. SplitStream [4] is a distributed implementation of this approach that is based on a locality aware DHT called Pastry [10]. SplitStream and systems alike have the advantage of being topologically aware (trees are formed according to the network distance between nodes) leading to small setup time as the propagation of a block from the root of the tree to the leaf nodes is done through nodes which are physically close in the underlying network. However these systems suffer from two main drawbacks: a) they don't take into account the heterogeneous upload capacities of the peers [17], and b) they can't cope up with the dynamic behavior of the participating peers as well as the underlying network as observed in commercial P2P streaming systems [18],. When a peer leaves the overlay, the path between it and its descendants is broken resulting in idle descendants during the reconstruction phase of the tree.

The second category described in [2],[12],[15]. Each node maintains connections with a relatively small number of nodes which are considered as its neighbors in the overlay. The overlay is constructed randomly or according to the upload capacities of the nodes that participate in it. Blocks that are generated by a server have playback deadlines. Each peer exchanges and maintains a number of lists (buffers), one per neighbor. Each one of these buffers contains those blocks of its neighbor that their playback deadline has not expired yet. To this end, a peer is capable at any time of making a decision about which block should be transmitted to which neighbor. This decision process is implemented by a scheduler running in every node. The characteristic of these systems is that the block transmissions are agnostic to the overlay topology.

Due to their architecture the main advantage of them is their flexibility which allows them to take advantage of the heterogeneity of the participating peers and deal with the dynamic behavior of the system leading to higher levels of bandwidth utilization. However, these systems can't exploit the network proximity among the peers that exchange blocks. This means that the time required for a block to be transferred from one node to another and hence the required time for all nodes to acquire the block (setup-time) could reduced if the overlay exploits the locality between peers. Another drawback of overlays agnostic to locality is that buffer exchanges between neighbors performed with high network latency. This effect leads duplicate block transmissions and so to wasted upload bandwidth.

The primary contribution of this paper is the creation of an overlay where each node discovers and exchanges blocks with the nodes physically close to it in the underlying network. The physical network distance is captured by means of a novel, locality aware structured P2P graph which is reconfigured dynamically according to the latencies between nodes in the underlying network. This overlay can approximately be seen as a self-organized d-dimensional grid where the position of each node in the overlay reflects its position in the underlying physical network. There are two algorithms that every node runs. The first is the placement algorithm that each node runs only once when it enters the system and is responsible for finding a suitable neighborhood of the overlay for this node. The second one, called the swapping

algorithm, aims at the distributed and dynamic optimization of the neighborhoods in order to reflect the underlying network.

The rest of this paper is organized as follows. In section 2 we present our proposed overlay and we describe in detail the placement and the swapping algorithm in Section 3 we describe briefly the scheduler that we use in order to evaluate our overlay and in Section 4 we evaluate its performance. At last in section 5 we conclude and we point our future work.

2 The Proposed System

2.1 Locality Aware P2P Overlay Architecture

For creating a locality aware P2P overlay where each node has as neighbors the nodes in the underlying network that are physically close to it, we have used structured P2P overlay, in particular we have use the neighbor table maintenance mechanism from CAN [1], CAN is a distributed self-organized overlay, which intuitively approximates a d-dimensional grid. In CAN each node holds a random portion of a d-dimensional space. It offers three major advantages that explain our architectural decision to use it as a substrate for our streaming system. The first is that is guarantees that each node will have at least 2*d incoming neighbors and so at least 2*d nodes will provide blocks to it. The second is the balanced properties of the overlay (are demonstrated in the evaluation section) that means that there are no nodes with a large number of outgoing neighbors and so as we will see later that feature reduces the control overhead of live streaming. The third and most important feature of this overlay is that between any two nodes there are at least 2*d paths where in each one a different set of nodes participates. Given this attribute and by the creation of the appropriate scheduler no point of the graph is a bottleneck in live streaming. At last offers a mechanism to formulate our distributed optimization algorithm as we will analyze in section C.

In contrast, locality aware DHTs proposed so far, lead to overlays where nodes have highly unbalanced number of neighbors [8] and so are unsuitable to be used as overlays in P2P streaming.

In order to reflect the underlying network in the d-dimensional space of CAN, thus capturing locality, we have developed two algorithms. The first, influenced from [16], is called *placement algorithm*. This is responsible for navigating and placing each node that enters our system next to its closest node in the physical network. The second called *swapping,* is responsible for the distributed and dynamic optimization of the overlay according to the network latencies between nodes. These two algorithms constitute the extension of the original CAN leading to a locality aware overlay called L-CAN.

2.2 The Placement Algorithm

The placement algorithm is a distributed algorithm responsible for finding a suitable neighborhood for each node which enters the system. Its high accuracy is not a critical issue, because we do not rely entirely on this algorithm to provide an optimal placement of the nodes (neighbors that are also physically close to each other) in the

L-CAN. This is the job of our swapping algorithm that is responsible for an optimal dynamic and stable solution.

Each node, already part of the overlay, must be capable of navigating each new node closer to its final position. To do so, each node i in the L-CAN must maintain some kind of information about the structure of the overlay. The first source of information comes from the neighbors of i as through the exchange of control messages with them, node i can get an estimate of the corresponding STTs. In addition to communicating with its neighbors, each node communicates with a number of nodes R_i uniformly spread in all d dimensions of L-CAN (Figure 1). We call these nodes *ring nodes*.

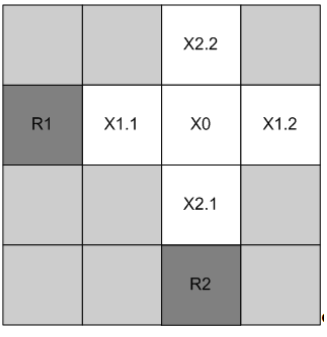

Fig. 1. Node X0 is already placed in a two dimensional CAN. Its neighbors are nodes {X1.1, X1,2, X2.1, X2.2} and communicates with one additional node in each dimension (R_i=1), namely, its ring nodes {R1,R2}.

Every node N, which is already placed in L-CAN, has a structure (called `topology_list`(N)) that holds the values of the estimated STTs with its neighbors and its ring nodes. Methods and algorithms which provide STT estimates through many RTT measurements are described analytically in [21].

When a node X is to be inserted in our system it makes use of two lists. The first one, called `closer_list`, contains those nodes which our placement algorithm has found upon completion of an iteration to be closer to node X up to a maximum number `closer_num`. The second one, called `check_list`, is used and updated at every step of the placement algorithm in order to infer from the `topology_list` of those nodes in the `check_list` additional nodes that happen to be close to X. The maximum size of `check_list` is set to `check_num`. (Figure 2).

In order to bootstrap the placement algorithm, the new node X randomly selects a node N already in the L-CAN and places it in both lists. Then, the new node performs the following iterative steps until its `closer_list` remains the same for a number of `stop_check` steps.

For each node N_i in X's `check_list` the new node X takes a pre-defined number of nodes (`cl_num`) from N_i's `topology_list` which are potentially closer to X. This is done by calculating the absolute difference between the STT from N_i to X and the STT from N_i to the node which was present in N_i's topology list 16. Then X probes

these `cl_num*check_num` nodes and if it finds that some of these nodes are closer to X than those in its `closer_list`, it substitutes those nodes in the `closer_list` with the new nodes with smaller STTs. Finally, `check_list` is assigned the contents of the `closer_list` and the process is repeated until convergence.

When the algorithm terminates, the first node in the `closer_list` is the nearest to X in the physical network. The pre-defined numbers `closer_num`, `check_num`, `cl_num` and `stop_check` define the trade-off between accuracy and speed of convergence of our placement algorithm.

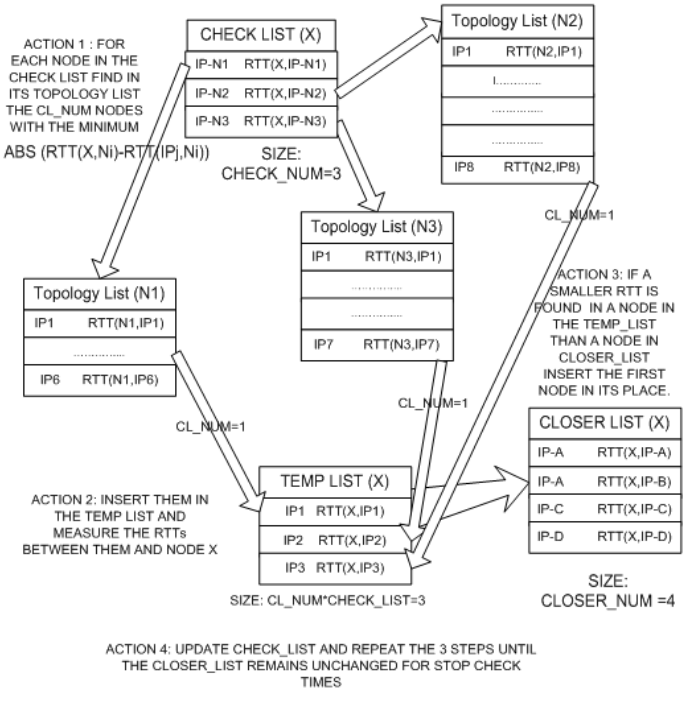

Fig. 2. At each step 4 actions are performed. The data structures which the new node X maintains during its entry in the DHT. N_i denotes the nodes which X asks at each step.

After the discovery of the physically closest node to X, node K, X sends to K a request for insertion. At this point we face another challenge: to create a balanced overlay, namely, the number of neighbors of every node must be similar in size. To this end, X checks whether a neighbor of K has a zone bigger than K's zone. If such a neighbor exists then the new node will be inserted in the overlay between K and the neighbor of K with the bigger zone, otherwise, it will be inserted in the zone of K. By doing this we still ensure that nodes K and X will also be neighbors in L-CAN, while we avoid the possibility of a node having a relatively big zone compared with another one close to it. In this way the requirement for a balanced overlay in terms of the size of neighbors is guaranteed.

2.3 The Swapping Algorithm

The swapping algorithm, aims at keeping L-CAN dynamically updated and optimized with respect to locality as opposed to the placement algorithm which is responsible for maintaining a balanced overlay during initial node insertion.

Two events may trigger the need for such a reconfiguration: a) the arrival or departure of a node, and b) any change in network conditions. Both events result in new STT values maintained by L-CAN, which, in turn, invokes the swapping algorithm to rearrange the "neighborhoods" in L-CAN.

The L-CAN swapping algorithm is based on a function, we call *energy function*, denoted as E(i,j), where i is a node in L-CAN and j is the zone that this node occupies, $i \in N \; and \; j \in Z$ where N,Z the sets of the nodes and the zones in L-CAN respectively. Additionally each zone j in L-CAN is adjacent with a set of zones noted as Zneigh(j) and the nodes that hold these zones defined as Neigh(i). Accordingly, the energy function of a node i that holds a zone j is defined as:

$$E(i,j) = \sum_{k=1}^{k=|Neigh(i)|} [Stt(i,k) \mid k \in \{Neigh(i)\}] \quad (1)$$

This small subset Neigh(i) of N is the nodes that i uses in order to exchange blocks. If we assume that links between neighbors are used with equal probability in streaming the performance of the streaming application is based on the metric:

$$E_{all} = \sum_{i \in N} [E(i,j) \mid \forall i1 \neq i2 \rightarrow j1 \neq j2] \quad (2)$$

If we minimize E_{all} the overlay would minimize the average latency and consequently it would also minimize the average probability for duplicates The algorithmic complexity of such problem is O(|N|!) provided that we apply an exhaustive search of all possible combinations among all nodes and zones of L-CAN. To this end, there is a need for a distributed algorithm that will minimize E_{all} or at least it will converge in a good sub-optimal value for E_{all}. So we propose an algorithm that dynamically rearranges "neighborhoods" in L-CAN. Accordingly our swapping algorithm starts from a node that we called as the initiator. In a swapping process the nodes that participate is the initiator and all of its neighbors. The goal of our swapping algorithm is an optimal assignment between these nodes and their zones that we denote with the set Nswap and Zswap respectively. As optimal assignment we define the assignment that results in the minimum sum of energy of the participating nodes in the swapping process. In order to reduce the complexity of such an assignment that is O(|Nswap|!) we formulate this process as a linear integer programming problem that solved with a polynomial complexity. In the rest of this section we describe this formulation in detail.

More specifically, Figure 3 illustrates an example of swapping among 5 neighboring nodes, Nswap={x0,x1,x2,x3,x4} that occupy zones Zswap ={p0,p1,p2,p3,p4} respectively, in a 2-dimensional L-CAN. x0 is considered the initiator of a swapping process.

Initially, nodes x0-x4 exchange with each other the set of their own neighbors. For instance, node X1 and X3 exchange Neigh(x1) = {a1,a7,a8,x0}, and Neigh(x3) = {a3,a4,a5,x0}, respectively. After all nodes that belong to Nswap have been notified of all these neighbor sets in each position, each one measures its STT in the nodes that

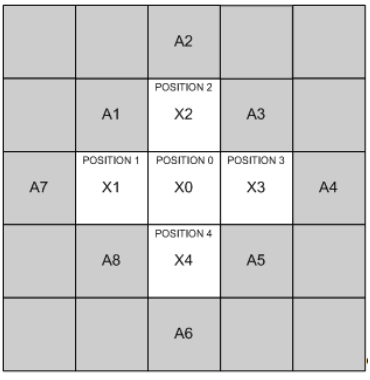

Fig. 3. A two-dimensional CAN where nodes {X0, X1, X2, X3, X4} perform a concurrent swapping, where X0 is the initiator

presented in every set Neigh(i) where i∈Nswap and send the measurements to the initiator as this is the node that carries out the execution of the swapping algorithm. Now the initiator is capable of calculating the optimal assignment between nodes Nswap and positions Zswap. In order to formalize the swapping process we note now as Zneigh(i) the set of nodes that are adjacent with a zone i and do not participate in the swapping process (Ex. Zneigh(P2) ={a1,a2,a3}).For the calculation of energy in every zone except the central one (P0 in our example) we use the set Zneigh this set excludes the node that will be moved to the central position as it is unknown yet. For instance, the energy of x1 to position P2 is calculated for the needs of the swapping as:

$$E(x1,p2)=Stt(x1,a1)+Stt(x1,a2)+Stt(x1,a3) \quad (3)$$

The problem with (3) is that we do not know which of the x0, x2, x3, and x4 moves to the central position P0. Four outcomes are possible, but we will describe later how we tackle this problem.

The case where a node moves to the central position P0, is simpler as each node of the calculation of its energy uses the set Nswap that participate in the swapping process except itself. That's because as we observe from the Figure 3 if this node moved to the central position will have as its neighbors these nodes. Again, if X1 moves to P0 its energy function becomes:

$$E(x1,p0)=Stt(x1,x0)+Stt(x1,x2)+Stt(x1,x3)+Stt(x1,x4) \quad (4)$$

We are now able to determine the energy of each node in a more abstractive way in order to formulate the swapping algorithm. In order to find the optimal assignment of the nodes∈ $Nswap$ and their zones∈ $Zswap$ we have to calculate the energy of each node in each potential position.

So for the zone of the initiator, noted as position 0, the node i that will be swapped there will have energy:

$$E(i,0) = \sum_{j=1}^{j=|Nswap|-1} [Stt(i,j) | \, j \in Nswap, i \neq j] \quad (5)$$

For the other zones k∈{Zswap-0} the energy of the potential node i that will be swapped there is:

$$E(i,k) = Stt(i,j0) + \sum_{j=1}^{j=|Zneigh|}[Stt(i,j)|\, j \in Zneigh(k)] \quad (6)$$

With j0 we note the node that will be swapped to the central position after the swapping process. As we observe the first term of this equation $Stt(i,j0)$ is independent from the position k and it depends only in the node that will take the position of the initiator that noted position 0. If we examine these factors we will observe that there will be |Nswap|-1 of them and each one expresses the Stt between the node that will placed in zone 0 and each node except it that belongs to Nswap. If now we observe E(i,0) we will find that it is equal with the sum of these factors as we observe from the equation 5. After this observation we have prove that by the selection of a node for the central position we determine a factor:

$$E'(i,0) = 2 * E(i,0) \quad (7)$$

While the selection of a node for the other positions k determines a factor:

$$E'(i,k) = \sum_{j=1}^{j=|Zneigh|}[Stt(i,j)|j \in Zneigh(k)] \quad (8)$$

After these calculations in order to perform the swapping process we want to minimize:

$$\sum_{i}^{|Nswap|}\sum_{j}^{|Zswap|} a(i,j) * E'(i,j) \quad (9)$$

Where a(i,j)∈ {0,1} and for each node i and each position j holds that:

$$\sum_{j=1}^{j=|Zswap|} a(i,j) = 1 \text{ and } \sum_{i=1}^{i=|Nswap|} a(i,j) = 1 \quad (10)$$

These are the constraints of the problem and intuitively express that each node will be placed in exactly one position and also in each position j will be placed exactly one node. As the nodes and the positions are equal this is an integer linear programming problem always feasible. After its solution for each a(i.j)=1 node i will be placed in position j. If it will moved in the central position t will take as neighbors the set {Nswap–i} and in the other positions j the set Zneigh(j) and the node that will move to the central position. This is a well know problem of linear programming and we refer to [20] for its solution.

There is an analytical proof that our algorithm converges but due to the restricted space we just describe that each time that the algorithm is executed to a neighborhood the energy of this neighborhood is reduced and so the total energy of the system. As a result of this property and the fact that the total energy is a positive number the whole system will converge to an assignment between nodes and positions will the low total energy and as we observe for our simulations it converges to a very attractive suboptimal.

3 P2P Live Streaming Systems

Without loss of generality we assume that in a P2P streaming system there is a bootstrap node which is used for the entrance 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. 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*.

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.

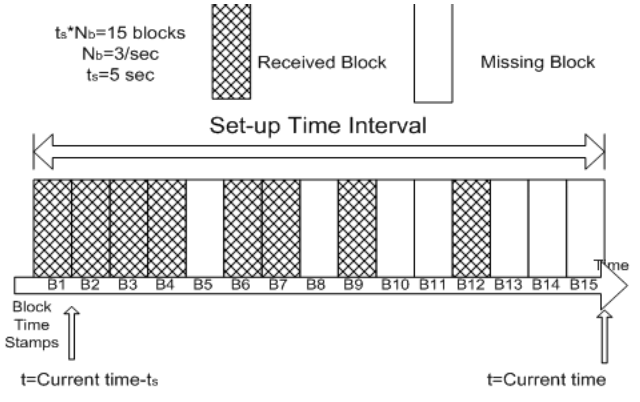

Fig. 4. Snapshot of a buffer in a node with the states of the blocks

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 4 provides a snapshot of the states of blocks of a buffer in a node.

More specifically, each node upon reception of a new block, propagates this information to all of its neighbors. Therefore, every time that a node wants to transmit a new block knows the blocks that it has and the blocks that their neighbors have. A scheduler, described in [2], proposes the transmission of a block to the neighbor that misses the largest number of blocks among those that the transmitting node has. According to it, each node i has to decide which neighbor j must serve first, by calculating the difference(i,j) of the blocks that each neighbor j misses and are present

in node i. The node with the largest difference is the one selected for block transmission while ties are resolved arbitrarily. Finally, the block to be transmitted is selected randomly with a uniform distribution.

4 Evaluation

For the evaluation of our P2P streaming system we have used OPNET Modeler [24] in order to avoid the imperfections of a custom made simulator. We have tested our proposed system under various underlying network topologies. Here we present its performance based on a topology from [5], where the provided round trip time measurements were gathered using the King method between globally distributed DNS servers. 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. In the rest of this section we present a system with 2000 nodes.

Before examining our system and its performance in live streaming, we present the results that show the cumulative density function of the number of neighbors that each node has for a random mesh and for our overlay. As we observe in Figure 5 our overlay is more balanced than a mesh although the nodes are placed according to their position in the underlying network. At this point we want to clarify that we don't claim that nodes have similar zone sizes (ex. In network regions with high density of nodes the zones of LCAN may be smaller) as we don't want to use L-CAN as a DHT in order to store keys in it. In contrast we want to have a balanced overlay in terms of the neighbors that each node has. We achieve this goal due to our placement algorithm as described earlier.

Now in Figure 6 we demonstrate the performance of our overlay in live streaming compared with the performance of a randomly created mesh. We examine two scenarios: a) all the participating nodes have homogeneous upload capacities, and b) the upload capacities of the nodes conform to those in [3] that have been collected from users of a real P2P system and they are heterogeneous.

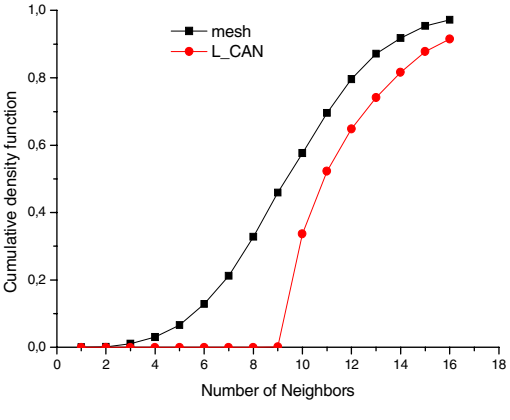

Fig. 5. Cumulative density function of the number of neighbors that each node has in a random mesh (black line) and in L-CAN (red line)

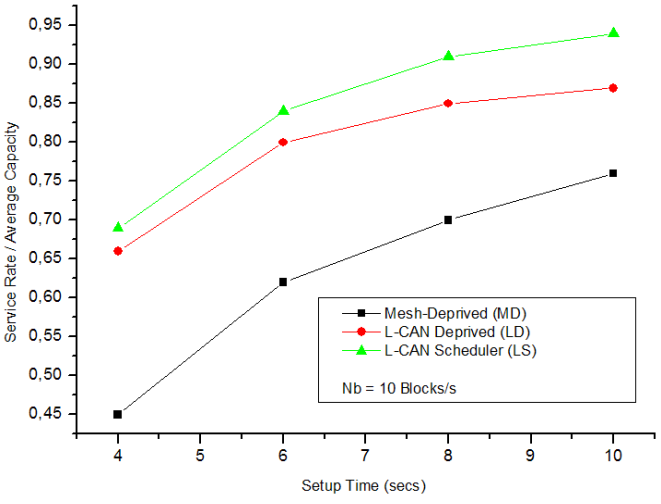

Fig. 6a. Maximum achievable service rate (as a percentile of the average upload bandwidth) of each system under various setup time intervals. Nodes contribute equal upload bandwidths.

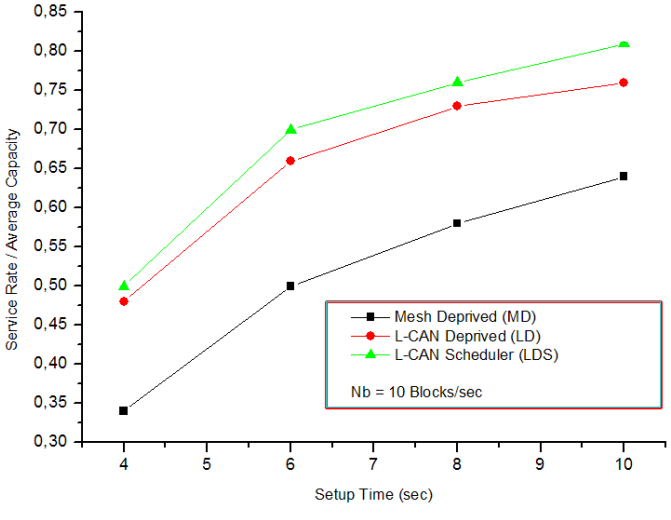

Fig. 6b. Maximum achievable service rate (as a percentile of the average upload bandwidth) of each system under various setup time intervals. Nodes contribute heterogenous upload bandwidths.

Based on these two scenarios we have evaluated three different systems. The first system, denoted as MD, is a mesh overlay where each node has 10 neighbors, while the exchange of blocks among nodes is performed according to the most deprived node scheduler [2] as described earlier. The second system, denoted as LD, uses the same scheduler as MD, but it relies on our topology aware overlay L-CAN with

5-dimensions (each node has approximately 10 neighbors) which replaces the mesh overlay. In the third system, denoted as LS, we have maintained the L-CAN overlay and we have substituted the scheduler in MD with our own.

This new scheduler we propose, though simple, exploits in its logic the existence of a locality aware overlay and in particular every node's likely network position reflected in the L-CAN. According to it, each node keeps an incremental list of its neighbors according to STT between them. The probability with which each one of its neighbor is selected for a block transmission is proportional to its rank in the list. Figure 6 shows the ratio between the maximum achievable service rate that each system can deliver and the average upload bandwidth of the participating nodes, for various setup time values given a constant rate for block generation (N_b=10 blocks/sec). For the first graph we have used homogenous upload capacities whereas the later is based on heterogeneous. Inspecting the two graphs in figure 6, we observe that the same performance trend emerges from either case i.e. homogeneous and heterogeneous upload capacities. Furthermore, all systems perform slightly worst in case of heterogeneous upload capacities.

As predicted by our analysis applying a locality aware overlay, L-CAN, results in a significant increase in the achievable service rate (LD), as opposed to a mesh overlay (MD), because of the smaller STT values that exist between the neighbors in L-CAN. Further improvement especially for not very small set-up time values is obtained when we apply our scheduler in (LDS) as we see observe from the blue line in figure 6.

Finally, the control overhead of our scheduler is trivial. Every time that a node receives a new block it sends to all of its neighbors a control packet with the contents of its buffer. The frequency of this transmission is equal to N_b by the definition of the scheduler. Additionally the size of the buffer is t_s*N_b and is modeled with one bit of information for each block (ex. One for its presence and zero for its absence). So the control bandwidth that is consumed in each node in order to update the other nodes for the new block arrivals is approximated by:

$$C.B.=(Header\_size+t_s*N_b)*N_b*Number\_of\_neigh$$

where $Header\_size+t_s*N_b$ is the size of each control packet Nb is the frequency that this is transmitted and Number_of_neigh is the number of neighbors in which the control packets are send. By the value of Nb=10 blocks/sec as, t_s=5sec and Number_of_neigh=10 the control bandwidth that consumed in each node is has a value around 30 kbps and its independent from the upload capacities. This means that with higher upload capacities of peers the overhead that is consumed from the buffer exchange lowers as a percentage of the upload bandwidth.

5 Conclusions and Future Work

As we have observed though our evaluation L-CAN greatly improves the set-up time and the bandwidth utilization in live streaming. Additionally combined with our previous work in [22] enables locality in CAN and the whole system can be used in other applications such as multi-attribute range queries. Furthermore we will modify and apply our algorithms in other overlays in order two evaluate them in these.

In live streaming our future work is focused in two major areas. The first is the creation of a sophisticated scheduler that exploits locality and the information that is infused in any node during the buffer exchanges by selecting the appropriate block for transmission and not only the appropriate node. The second is the selection of the appropriate number of neighbors and the adjustment of the block size in order to further improve the performance of our system. At last we plan to evaluate our system in dynamic network conditions to demonstrate its adaptive behavior.

References

1. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: ACM Sigcomm, San Diego (2001)
2. Massoulie, L., Twigg, A., Gkantsidis, C., Rodriguez, P.: Randomized decentralized broadcasting algorithms. In: 26th IEEE International Conference on Computer Communications (INFOCOM), pp. 1073–1081. IEEE Press, Anchorage (2007)
3. Bharambe, A., Herley, C., Padmanabhan, V.: Analyzing and improving bittorrent performance, Technical Report, Microsoft Research (2005)
4. Castro, M., Druschel, P., Kermarrec, A.M., Nandi, A.: Antony Rowstron, Atul Singh, SplitStream: High-Bandwidth Multicast in Cooperative Environments. In: SOSP, New York (2003)
5. Meridian, A.: Lightweight Approach to Network Positioning, `http://www.cs.cornell.edu/People/egs/meridian/data.php`
6. MIT Parallel and Distributed Operating Systerms Group, `http://pdos.csail.mit.edu/P2Psim/kingdata/`
7. Ng, T.E., Zhang, H.: A network positioning system for the Internet. In: USENIX Conference, Boston (2004)
8. Castro, M., Druschel, P., Hu, Y.C., Rowstron, A.: Exploiting network proximity in distributed hash tables. In: International Workshop on Future Directions in Distributed Computing (FuDiCo), Bologna (2002)
9. Stoica, I., Morris, R., Liben-Nowell, D., David, R., Karger, M., Kaashoek, F., Frank Dabek, F., Balakrishnan, H.: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. IEEE/ACM Transactions on Networking 11(1), 17–32 (2003)
10. Rowstron, A., Druschel, P.: Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)
11. Gkantsidis, C., Rodriguez, P.: Network coding for large scale content distribution. In: 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings (INFOCOM), pp. 2235–2245. IEEE Press, Miami (2005)
12. Kumar, R., Liu, Y., Ross, K.W.: Stochastic Fluid Theory for P2P Streaming Systems. In: 26th IEEE International Conference on Computer Communications (INFOCOM), pp. 919–927. IEEE Press, Anchorage (2007)
13. Zhang, X., Liu, J., Li, B., Yum, T.-S.P.: CoolStreaming: A Datadriven Overlay Network for Peer-to-Peer Live Media Streaming. In: 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings (INFOCOM), pp. 2102–2111. IEEE Press, Miami (2005)
14. Castro, M., Druschel, P., Hu, Y.C., Rowstron, A.: Exploiting network proximity in distributed hash tables. In: International Workshop on Future Directions in Distributed Computing (FuDiCo), Bologna (2002)

15. PPLive, http://www.pplive.com
16. Wong, B., Slivkins, A., Sirer, E.G.: Meridian: A Lightweight Network Location Service without Virtual Coordinates. In Proceedings of ACM SIGCOMM, Philadelphia (2005)
17. Magharei, N., Rejaie, R., Guo, Y.: Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches. In: 26th IEEE International Conference on Computer Communications (INFOCOM), pp. 1424–1432. IEEE Press, Anchorage (2007)
18. Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K.W.: A Measurement Study of a Large-Scale P2P IPTV System. IEEE Transactions on Multimedia 9(8), 1672–1687 (2006)
19. Magharei, N., Rejaie, R.: PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming. In: 26th IEEE International Conference on Computer Communications (INFOCOM), pp. 1415–1423. IEEE Press, Anchorage (2007)
20. Dimirti, P.: Bertskeas, Network Optimization: Continuous and Discrete Models. Athena Scientific (1998)
21. Pietzuch, P., Ledlie, J., Seltzer, M.: Supporting Network Coordinates on PlanetLab. In: Proceedings of WORLDS (2005)
22. Efthymiopoulos, N., Christakidis, A., Denazis, S., Koufopavlou, O.: Enabling locality in a balanced peer-to-peer overlay. In: IEEE Global Telecommunications Conference (GLOBECOM), San Fransisco, pp. 1–5 (2006)
23. Hei, X., Liu, Y., Ross, K.W.: Inferring Network-Wide Quality in P2P Live Streaming Systems, Technical Report (2006), http://eeweb.poly.edu/faculty/yongliu
24. Opnet Technologies, http://www.opnet.com