Analysis of Block-aware Peer Adaptations in Substream-based P2P

Chamil Kulatunga, Dmitri Botvich, Sasitharan Balasubramaniam, William Donnelly
Telecommunications Software and Systems Group, Waterford Institute of Technology
Cork Road, Waterford, Ireland
{ckulatunga, dbotvich, sasib, wdonnelly@tssg.org}

Abstract: **Peer-to-Peer (P2P) video delivery using substreams supports uplink heterogeneities of the peers and hence could optimise sharing capabilities with minimum free-riding peers. Therefore, substream-based applications such as PPLive and CoolStreaming have been well accepted after successful deployments in the public Internet. In this approach, a child peer can find a parent peer for a substream independent of the other parent peers that it receives the remaining substreams. In general, there can be more than one substream between a parent and a child. The block-aware adaptation algorithm in CoolStreaming changes the parent peer for all such substreams when a child peer experiences poor performance even on one of its substreams from the parent. However, lagging of one substream in such a scenario is likely while others are not affected, when the parent receives its substreams through multiple paths. We propose a fine-grained approach (changing substream by substream) in peer adaptations to improve overlay network performance. This approach will in turn, is designed also to minimise the diversity of parents at the child peer by joining with a well-performing another parent, which is expected to curtail complexities in a network-assisted P2P framework.**

Keywords: Video streaming, substream-based P2P, child-initiated block-aware peer adaptation.

I. INTRODUCTION

In recent years, Peer-to-Peer (P2P) multimedia streaming has gained increased popularity, due largely to its scalable solution for video streaming to a very large number of concurrent users. P2P file sharing was technically unbeaten mainly due to its flexibilities of distributing different amounts of data *blocks* (chunks) (i.e. due to its non-real-time application requirements) [1], where a peer can reliably collect the required set of blocks of a file from any number of peers within a reasonable time frame, disregarding the order of the blocks. However, this is not the case in P2P multimedia streaming, where playback delay and its continuity become vital Quality of Experience (QoE) factors. In the absence of the flexibilities available with file sharing, using the same approach for streaming video applications will face a number of serious challenges [2] [3]. A number of solutions have been developed to counter these challenges, such as applications like PPLive, CoolStreaming, SopCast, Babelgum, which are currently been deployed in the public Internet with a marginal streaming quality (i.e. bandwidth in a range of 300 to 500 kbps and playback delays of 10s to several minutes).

Many P2P streaming protocols use a hybrid push-pull approach to avoid instabilities of a tree-based push overlay structure owing to deep trees [4]. Pulling capability can be implemented replicating the same stream into multiple trees so that a peer can pull the stream from any tree which will improve its overall performance. However, this approach leads to unnecessary replication of data in the network and does not support uplink heterogeneities. A solution to this problem is to sub-divide the main stream into a set of *substreams* (also known as substreamed P2P). In order to collect all the substreams (i.e. an essential requirement without SVC or MDC), a peer is required to join with multiple (but low-bandwidth) trees. This hybrid push-pull technique has become a victorious approach in deploying P2P video delivery over the public Internet especially with asymmetrical residential peers like ADSL to improve sharing capabilities minimising free-riding peers.

A prominent example of substreamed P2P is CoolStreaming. CoolStreaming [6] *peer adaptations* (i.e. the process that a peer selects a new parent during the session when a substream performance is degraded) are triggered by a child, which we categorise as a *child-initiated* process. The approach uses two inequalities (i.e. for testing the performance between the child and the parent and between the parent and the other partner peers). The only performance metric used by a child during the selection process is recording and comparing the latest received block at each substream. We categorise this also as a *block-aware* approach. Peer adaptations in non-substreamed P2P do not need to differentiate performances in source-to-parent or parent-to-child paths, since the entire session is received along a single path at a time. The only solution is to change the parent irrespective of the location of the tarnished performance. It also has no flexibility of responding differently for peer dynamics (i.e. peer-churn) and network dynamics (i.e. congestion). However, in substreamed P2P, parent changes can be done independently from one parent to another or from one substream to another (one child may have multiple parents and one parent may deliver more than one substreams to a child) and can respond flexibly.

In this paper, we analyse and evaluate CoolStreaming peer adaptation algorithm, and propose a new algorithm that extends from CoolStreaming to capture the above mentioned criterions and flexibilities. The original CoolStreaming forces a child to change all the substreams from a parent even though only one substream is under-performing. However, in our proposed algorithm, we avoid changing all substreams where only the under-performing substream is required to find a new

parent. We also propose removing one substream at a time by the child peer (i.e. a fine-grained conservative approach), if the identified congestion is in the uplink of the parent (i.e. expecting an improvement of congestion due to the granted space like in congestion control mechanisms).

The proposed solution also aims to minimise the diversity of parents (i.e. the number of parents that a child is required to acquire all the substreams) at a child peer without degrading the performance, which is expected to minimise overhead in a network-assisted P2P framework. This is achieved by joining with a well-performing another parent before seeking a new parent. We believe that since CoolStreaming will change all substreams when one of the substream is underperforming, this could lead to instabilities of the P2P network.

The rest of the paper is organised as follows: Section II provides related works. Section III details the child-initiated block-aware peer adaptation algorithm and discusses the capabilities and complications. The proposed fine-grained algorithm is presented in Section IV. Section V provides the simulation results and Section VI concludes the paper.

II. RELATED WORKS

P2P protocols like NICE [7] and ESM [8] use a tree-based approach for video streaming, which were first proposed as an alternative to solve infrastructure requirement of native IP multicast in group communications. This approach was initially thought to be the most suitable for streaming video when compared with mesh-based approaches [9], which was successfully used for file deliveries. The tree based approach supported low latency and low per-block overhead for long-lived streaming applications. However, many peers in a single tree topology were leaf-nodes, which did not contribute for data forwarding (only acted as data consumers). A peer-churn by an upper level node of the tree, in turn simultaneously affected a large number of nodes (i.e. especially when the tree depth is large) mounting instabilities of the overlay network. In order to minimise the above mentioned problems, the single tree-based streaming delivery approach has been extended to support multiple trees. AnySee [10] supports replication-based multi-tree approach. However, it does not support uplink heterogeneities. SplitStream [11], ChunckySpread [12] and mTreebone [13] principally introduced the substreaming approach without putting much attention on block-based video deliveries and the peer adaptation algorithm.

The substreamed approach has been practically deployed in the Internet by PPLive [14] and CoolStreaming [6]. It has been followed by recent works of P2P streaming as a solution to address the network dynamics and mutual contributions successfully [15]. CoolStreaming is the one which has published its peer adaptation algorithm. Zhenjiang Li et al. [18] has recently analysed the substream scheduling problem using max-flow model. It is important to further extend research works on performance optimisation of push-pull based substreaming algorithms for video delivery in the public Internet and in particular analyse the peer adaptation algorithms in detail.

III. CHILD-INITIATED BLOCK-AWARE PEER ADAPTATIONS

In substreamed P2P, the source divides the main video stream into equal video blocks (e.g. with a one second play time) and delivers into *N* number of substreams. These blocks are assigned to substreams in revolving fashion. The receivers are required to collect all the substreams from at most *N* number of parent peers and reorder them according to the block number so that it can be played back with minimal disruption. In the event that any block misses the playback point, the video is discontinued. A substream can lag due to a peer-churn or slow data-rate (due to congestion) in the last hop or above. In such a case the child peer can change the parent peer after exceeding a threshold specified in number of blocks.

In such a peer driven adaptation algorithm, a peer needs to know the *block-maps* (i.e. the list of latest received block number of each substream) of its own (C), its parent peers (P_i) and other partner peers (Q_i) those it can select to join. To maintain the scalability of the protocol, a peer will only exchange block-maps between a selected number of partner peers (among them at most *N* could become parents). They periodically exchange updated block-maps using a *gossip algorithm* [6]. There can also be other peers (besides P_is and Q_is) which are members (M_is) of the session without having any interaction with an identified child peer (peer C in Figure 1).

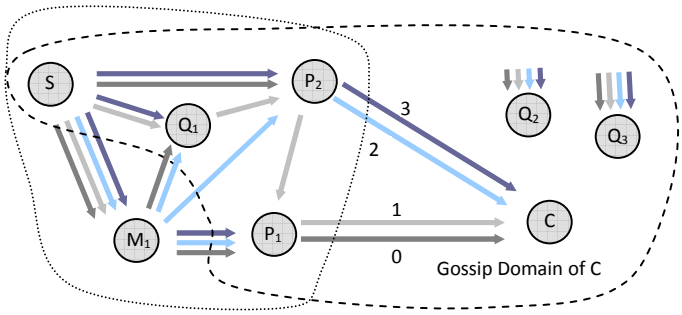


Figure 1. An overlay network with 4 substreams

In Figure 1, we assume that P_1, P_2, Q_1, Q_2 and Q_3 are the partners of child peer C and at present it receives the substreams 0 and 1 from parent P_1 and substreams 2 and 3 from parent P_2. If the current parent peer is needed to be changed, it will find a better parent (if the received block-maps of Q_1, Q_2 or Q_3 is better that P_1 or P_2, will connect to Q_1, Q_2 or Q_3) to receive the substreams.

A. CoolStreaming Peer Adaptations

According to the peer adaptation approach used in CoolStreaming, a child peer will use two inequalities (given in the equations *1* and *2* [6]) to identify a requirement to change a parent for a substream *j* (*j* = 0 .. *N*-1). Satisfaction of either one of the inequalities will lead to a change in the parent.

$$\max\{|B_{i,C} - B_{j,P}| : i \le N\} < TH_C \quad (1)$$

$$\max\{B_{i,Q} : i \le N, Q \in Partners\} - B_{j,P} < TH_P \quad (2)$$

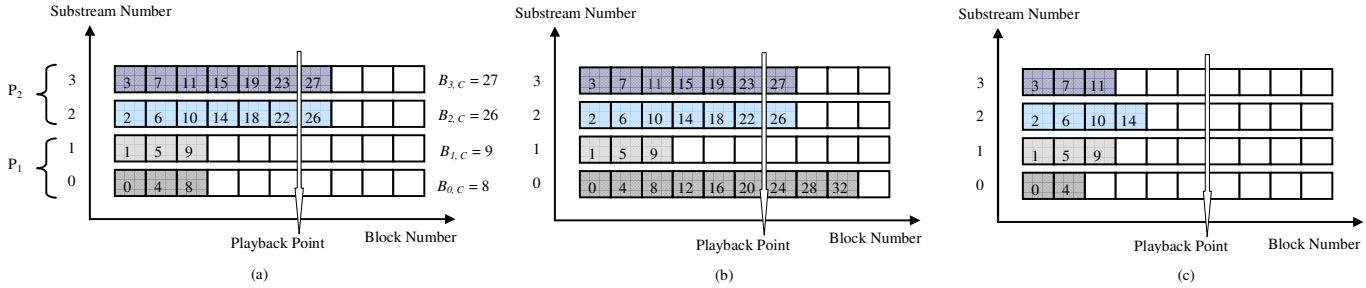


Figure 2. Parent change triggering situations at a child peer

$B_{i,X}$ is the latest received block for substream i at node X, where X could be either a child node C, parents P, or partners Q. TH_C is the threshold of the maximum deviation of latest received blocks allowed between the substream j at the parent and any substream at child node C. TH_p is the threshold of the maximum deviation of blocks allowed between the substream j at the parent and any substream at any partners.

These two tests are carried out periodically for all the parents at a child peer. The significant factor here is that if any substream (when receiving more than one substream from a parent) lags, CoolStreaming algorithm changes the parent for all the substreams originated from the same parent. This process will lead to finding a new parent peer, which also satisfies the inequality (*1*).

B. Analysis of Triggering Events

In order to analyse the algorithm, we consider three distinguished generic substream lagging situations (Figure 2), which could trigger a peer adaptation at a child peer. In the first case (a), both substreams from parent P_1 lag behind others. In the second case (b), only one substream (i.e. substream 1) from parent P_1 lags (this is possible since parent peer P_1 may receive two substreams from two different routes; 0 through M_1 and 1 through P_2 in Figure 1). In the third case (c), all the substreams are below the playback point.

Substreamed P2P is also a candidate transport mechanism that is compatible with recently accepted (by the IETF) multi-path TCP (MP-TCP) [5], which paves a path for resource pooling in the Future Internet. MP-TCP load balances a session in transport layer through the available interfaces in a multi-home environment. Since the content layering is inherited in substreamed P2P it can effectively be used over multi-path congestion control mechanisms. In such a scenario a parent peer may have performance differences between substreams even thought they are receiving from the same upstream peer. Therefore decedent child peers need to identify this situation in the peer adaptation algorithm, which will be in case (b).

The factors that may affect the conditions in Figure 2 may result from peer-churns or congestion bottlenecks in the core or access networks in the Internet. However, according to common analysis in P2P overlay networks, congestion is only considered in the uplink or downlink of a peer. We use the same assumptions in this analysis. We also assume that a peer-churn of an immediate parent can explicitly be identified by the child (may be using *ping*). Therefore peer-churn of such a parent (either P_1 or P_2 for child peer C in Figure 1) has not been considered under the triggering events discussed in the following paragraphs.

Each situation for peer adaptation (in Figure 2) results in several events shown in Table I, due to differing peer-churns or congestion in divided end-to-end overlay path; source-to-parent (multiple hops) and parent-to-child (last hop). In the table, L represents *Low* and H represents *High* in terms of the maximum available block at each substream. There can't be H at a child while having L at the parent since child can only acquire the data available at the parents. Theoretically, it is also not possible to have a situation where the parent's condition is H and child having L while another substream between the same pair of peers staying at H. We assume all the substreams between two peers follow the same path (if MP-TCP is used at a child, it is known to the peer and can remedy this situation) and experience the same congestion.

There can be two events between parent peer P_1 and the child peer C under the situation (a). The reason for case I to happen is when the parent P_1 receives delayed substreams from the source (i.e. due to either a peer-churn or congestion from source to parent). Solution for I is to change both substreams away from the current parent peer as quickly possible. The reason for case II could be due to congestion in the uplink at parent P_1. In our proposed solution, we would change one substream at a time rather than all used in CoolStreaming. This would, therefore, allow space for other substreams to grow.

TABLE I. Permutations for different triggering situations

Substream →		0		1		2		3		Congestion
Event ↓		C	P_1	C	P_1	C	P_2	C	P_2	
(a)	I	L	L	L	L					Source-Parent
	II	L	H	L	H					Parent-Child
(b)	III	H	H	L	L					Source-Parent
(c)	IV	L	H	L	H	H	L	H	L	Parent-Child
	V	L	L	L	L	L	L	L	L	Source-Parent

L-Low, H-High

The occurrence of case III is certainly due to poor performance above the parent P_1 since substream 0 does not show any performance degradation. According to Figure 1, this

can happen due to congestion between peers P_1 and P_2 or peer Q_1 leaving the overlay network. Therefore, only the lagged substream should be changed immediately. In CoolStreaming, the child unnecessarily changes the parent for both substreams due to the lagged substream 1.

The reason for case IV should be congestion in the downlink of child peer C or simultaneously in uplinks of both parents P_1 and P_2. We will follow the conservative approach by removing one substream from each parent. If this does not improve the performance, then the congestion in child peer's downlink maybe the factor resulting in poor performance. Case V arises when all substreams to the parents are delayed, in which case all substreams should be switched to new parents.

IV. FINE-GRAINED SUBSTREAM CHANGE

This section describes the steps for the proposed fine-grained approach, which considers changing substreams more conservatively. Performance of each substream is tested independent of other substreams, even though they originated from the same parent.

A. Conservative Algorithm

The first step of the algorithm is to identify the most lagged substream (j) for a parent (l) among N_l number of substreams received from the selected parent ($N_l \leq N$). The algorithm will then test for inequality (*3*), and determines if the deviation from the most progressed substream (among all the N number of substreams of the session) has exceeded the defined threshed (TH_C).

$$\max\{|B_{i,C} - B_{j,C}| : i = 0...N-1\} > TH_C \quad (3)$$

If this condition is satisfied, then the child identifies the location of the problem (in source-parent or parent-child paths) using the bit-maps received. The maximum block of substream j at the child is compared with the maximum block of the substream at the current parent using inequality (*4*).

$$B_{j,P} - B_{j,C} > TH_C \quad (4)$$

If this condition (*4*) is satisfied, this means the parent's quality performance is good and the congestion is between the parent and the child. This will lead to a change of one substream, which has the least performance at the parent (if there is more than one substream from that parent). However, when selecting a new parent, the selection does not necessarily ensure that it is better parent than the existing one, where the selection will find a parent which satisfies the inequality (*4*). The sole objective is to change the path from the current parent. If the inequality (*4*) is not satisfied, this means there is no performance issue along the path from the existing parent to the child. The selected substream may have already received substantial delay at the parent. Therefore, it checks the comparative performance of the current parent with the other partners according to the inequality (*5*).

$$\max\{|B_{j,P} - B_{j,Q}| : Q \in Partners\} > TH_P \quad (5)$$

In contrast to the previous parent selection, in this case the substream changes the current parent only if a better partner is found. Otherwise it will continue with the current parent. Then the test (*4*) should be applied independently for all the remaining substreams of the selected parent and change the parent, if required.

Algorithm 1. Fine-grained Peer Adaptation Algorithm

```
for l = 0 … number of parents (L)
|   find the most lagged substream (j) among N_l ;
|   if ( MAX | B_i,C – B_j,C | > TH_C : i = 0 … N-1)
|   |   if ( B_j,P - B_j,C > TH_C )
|   |   |   function-X ( );
|   |   else
|   |   |   if (MAX | B_j,P - B_j,Q | > TH_P : Q All Partners)
|   |   |   |   function-Y ( );
|   |   |   end
|   |   end
|   else
|   |   if ( B_j,C - PLAYPOINT < TH_V )
|   |   |   if ( B_j,P - B_j,C > TH_C )
|   |   |   |   function-X ( );
|   |   |   else
|   |   |   |   if (MAX | B_j,P - B_j,Q | > TH_P : Q All Partners)
|   |   |   |   |   function-Y ( );
|   |   |   |   end
|   |   |   end
|   |   end
|   end
end
function-X ( )
|   remove 1 substream having least B_m,P : m = 0 …N_l ;
|   if (L > 1)
|   |   find an existing parent having the most number of substreams;
|   |   check that its own substreams does not need a peer adapt;
|   else
|   |   find a parent satisfying inequality (4);
|   end
end
function-Y ( )
|   if (L > 1)
|   |   find an existing parent having the most number of substreams;
|   |   check that its own substreams does not need a peer adapt;
|   else
|   |   change j to a new parent satisfying inequality (5);
|   end
|   test for other m values of the parent l
end
```

If test (*3*) is not satisfied, this means there is no much deviation between the best and the worst substreams. This can happen in two situations: all the substreams are good or all are bad. If all the substreams are much ahead of the playback point, we need to avoid any parent change. Therefore tests (*4*) and (*5*) will be applied only when the most lagged substream is

less than a threshold (TH_V) of the playback point. Then the same procedure is applied similarly for the other parents.

Downlink congestion: If all the substreams lag TH_V threshold, it could also be due to downlink congestion at the child. Therefore, the child memorises this peer adaptation. If the situation is not rectified after a certain number of attempts, the child will extend the cool-down time (the time duration that a child peer will not test for parent changes again) of peer adaptation to minimise unnecessary events (alternatively the child could also use multi-path transport).

If congestion is in the uplink of the parent peer, CoolStreaming child finds a new parent for all the substreams it receives from that parent. Also if there is more than one child at this parent, it will end up loosing all the child peers when triggering events come closely. This could add extra overhead to the parent. Therefore, one approach to minimise this is to synchronise triggering events under one parent and ensure they don't come too close to each other. However, synchronisation of triggering events may not be required in the fine-grained approach since it uses a conservative substream changing process.

B. Minimising the Diversity of Parents

It has been widely accepted that next generation P2P is an ISP-assisted network service. IETF is standardising a framework for this purpose called Application Layer Traffic Optimisations (ALTO) [16]. Here, a content provider needs to register with the ALTO service (owned by an ISP) to avoid throttling their P2P traffic. Through negotiations with the ALTO server, a peer can select its parent peers. Hence an ISP can enforce different policies like restricting traffic to its own network or local geography.

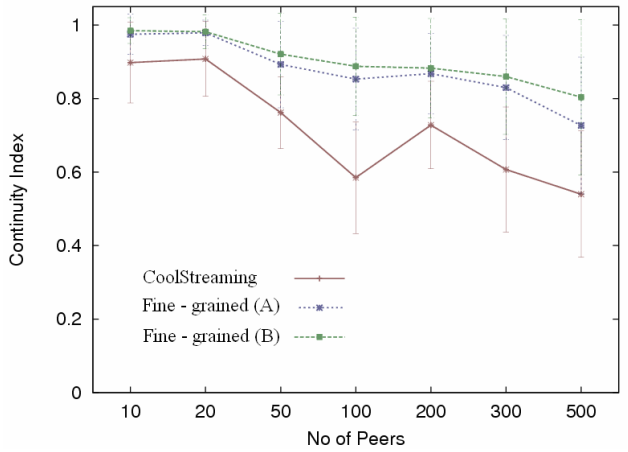
Although the standardisation through IETF is attractive and further increases the potential of P2P streaming, current approaches such as that used in CoolStreaming does not aim to minimise its complexities. For example, substreamed P2P may introduce an extra load on the ALTO server when requesting new parent peers for every single substream. Therefore, it could be desirable to minimise the diversity of parents at a child peer. This will improve self-organising capability in an ALTO domain reducing cross traffic (that will cost ISPs compared with local traffic) and also overhead at an ALTO server.

The fine-grained approach that we have proposed in this paper will minimise this effect, where we introduce a seeking process for new parents for a substream among existing parents. Therefore, when a substream is required to find a new parent, it will first seek a parent, which already delivers a substream to the child (that substream should not look to change the parent). If there are more than one qualified parents, then it will select the one delivering most number of substreams. If no other qualified existing parent is found, it will seek a parent from a larger partner list. Algorithm 1 presents the pseudo-code of the fine-grained algorithm including the parent diversity minimisation extension.

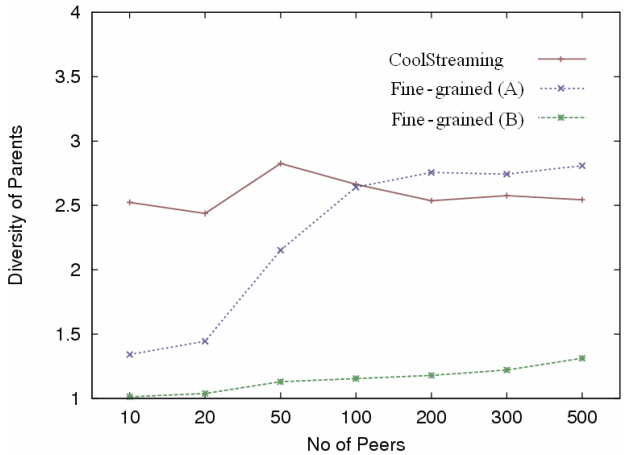
V. PERFORMANCE EVALUATIONS

We have simulated the CoolStreaming peer adaptation algorithm and the proposed fine-grained extension for two approaches; (A) seeking a new parent from the partner list, and (B) seeking a parent among the existing parents that the child is receiving other substreams, using OMNet++ simulator [17]. These algorithms were evaluated under a generic traffic model and a network topology. The time scale of a long-lived session has been contracted proportionately only to evaluate the peer adaptation process.

In all experiments, we used a traffic stream of 400kbps and it was divided into 4 substreams. The chunk size used for the substreams was 50 Kbytes, which is equivalent to a play time of 1s. The number of partners (those a peer was communicating) was limited to 5. Peer adaptation thresholds were chosen as; TH_C = 20, TH_P = 16 and TH_V = 0 in blocks. The cool-down time was 30s.



(a) Continuity Index



(b) Diversity of Parents

Figure 4. Performance with different number of peers

The uplink bandwidth at the server was 4 Mbps and at a peer it was randomly and uniformly distributed from 100 kbps to 1300 kbps in 100 kbps steps. This created a 400 kbps of average overlay uplink capacity on a participating peer (which

has been changed in the second set of experiments). Traffic was not limited at any other location in the overlay network than the uplinks. Background traffic was changed at uplinks randomly in 20s intervals uniformly distributed between 0 to 600 kbps, and again in 100 kbps steps.

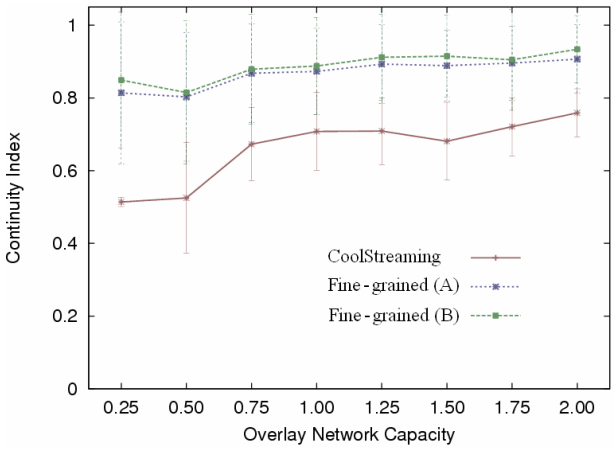
Peers joined randomly to a simple network topology. In order to simulate a heterogeneous substreaming scenario at a child peer, one third of the peers joined (i.e. at the start of their session) only with one parent for all the substreams. Then the subsequent one third of peers joined with totally different parents for each substream. The remaining peers joined with two parents with two substreams from each.

We measured QoE at a peer in terms of block continuity index (i.e. the number of blocks received at the playback point over total number of blocks it should receive). If one block misses the playback pointer, it backed-off 12 blocks rather continuing with the following block. We also monitored the diversity of parents at a child peer.

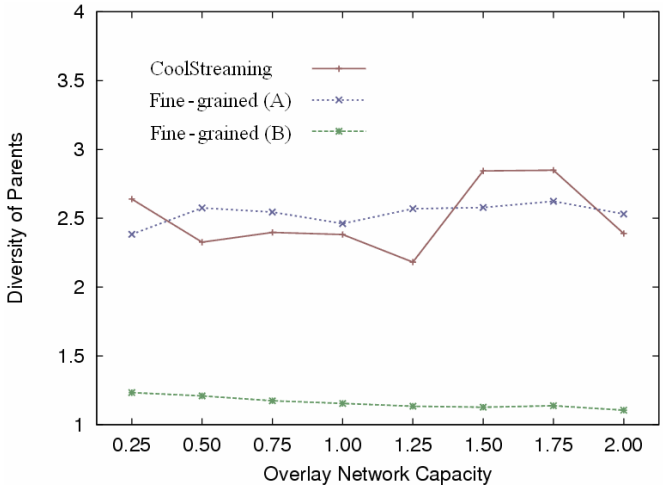
We conducted all the experiments for a duration of 1000s. Half of all the peers continuously connected to the overlay network for the entire duration. Remaining half created peer-churns by leaving the overlay for a duration between 0 and 20s at a randomly selected time. We have monitored the performance matrices at a child peer in 10s intervals and the average values of all the peers are shown in the following graphs.

We have first simulated algorithms with different number of peers. According to Figure 3 (a), a significant improvement of the continuity index can be seen with both proposed evolutionary approaches against the CoolStreaming peer adaptation algorithm (90% confidence intervals are shown in the graph). Figure 3 (b) shows that the diversity of parents is lesser in the evolutionary algorithm (A) with a number of nodes less than 100. However, the diversity increases as the number of nodes increases. Evolutionary algorithm (B) has notably reduced the diversity of parents.

We have then simulated three algorithms under different overlay network capacities. The average network capacity of all the uplinks was selected as a proportion to the full stream bandwidth requirement (i.e. in Figure 4, x-axis 2.00 indicates that the average uplink capacity is 800 kbps, which has network over-provisioned). The number of peers in these experiments was 100. According to Figure 4 (a), there is a consistent improvement of the continuity index using the two evolutionary algorithms. The diversity of parents has also not been affected much in evolutionary algorithm (A) but drastically reduced in the evolutionary algorithm (B) as shown in Figure 4 (b).



(a) Continuity Index



(b) Diversity of Parents

Figure 5. Performance with different overlay network capacities

Figure 5 shows the behaviour (from the start to the end of a session) of the continuity index at 20 randomly selected peers. According to the snapshot graphs and our observations, the continuity index approaches 1.0 and becomes steady during the entire session under the evolutionary approaches.

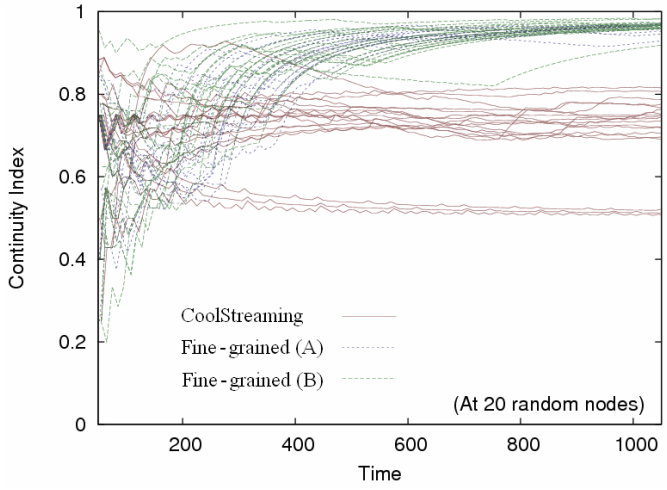


Figure 6. Change of Continuity Index (at 20 selected peers)

VI. CONCLUSIONS

P2P networking paradigm has been recognised by the IETF to re-design as a non-aggressive and ISP-friendly network service for the Internet. At the same time, P2P streaming will be used to solve future Internet bandwidth demands by federating core network resource requirements. Substreamed P2P is an important concept to support heterogeneous uplink bandwidths of residential peers and hence to improve co-operative resource sharing at the same time. Therefore, substreamed P2P concept needs to be developed while attempting to improve user's QoE.

In this paper we have proposed a fine-grained approach for the child-initiated block-aware peer adaptation algorithm that extends from the CoolStreaming application. The proposed approach utilises inter-substream performance parameters to differentiate source-to-parent and parent-to-child congestion and hence conservatively respond to changes in substream performance. The proposed solution also aims to minimise the diversity of parents, which could be problematic with the new network-assisted P2P standardisation initiative proposed by the IETF. Simulation results have been evaluated to compare the proposed solution with CoolStreaming, and the results have shown considerable improvement in QoE. We also claim that the new approach has minimised the diversity of parents.

ACKNOWLEDGEMENT

This work was supported by the project "FutureComm: Serving Society" funded by Higher Education Authority (HEA) in Ireland under the PRTLI scheme.

REFERENCES

[1] V. Aggarwal, A. Feldmann and C. Scheideler, "Can ISPs and P2P users Cooperate for improved Performance?", ACM CCR, July 2007

[2] J. Liu, S. Rao, B. Li and H. Zhang, "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast", Proceedings of the IEEE, January 2008

[3] H. Liu and G. Riley, "How Efficient Peer-to-Peer Video Streaming Could Be?", IEEE CCNC, January 2009

[4] X. Hei, Y. Liu, K. W. Ross, "IPTV over P2P Streaming Networks: the Mesh-Pull Approach", IEEE Communications Magazine, February 2008

[5] A. Ford, C. Raiciu, S. Barre and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", IETF Internet Draft, February 2010

[6] X. Zhang, J. Liu, B. Li and T. Yum, "CoolStreaming/DONet: A Data-driven Overlay Network for Efficient Live Media Streaming", IEEE INFOCOM, March 2005

[7] S. Banerjee, B Bhattacharjee and C. Kommareddy, "Scalable Application Layer Multicast", ACM SIGCOMM, August 2002

[8] Y. Chu, S. Rao, S. Seshan and H. Zhang, "A Case for End System Multicast", IEEE Journal on Selected Areas in Communication, October 2002

[9] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, F. Kaashoek, F. Dabek and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications", IEEE/ACM Transactions on Networking, February 2003

[10] X. Liao, H. Jin, Y. Liu, L. M. Ni and D. Deng, "AnySee: Peer-to-Peer Live Streaming" , IEEE INFOCOM, April 2006

[11] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: High-bandwidth Multicast in Cooperative Environments", ACM Symposium on Operating Systems Principles, October 2003

[12] V. Venkataraman, K. Yoshida and P. Francis, "Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast", IEEE ICNP, November 2006

[13] F. Wang, Y. Xiong and J. Liu, "mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast", IEEE ICDCS, June 2007

[14] X. Hei, C. Liang, J. Liang, Y. Liu and K. Ross, "A Measurement Study of a Large-Scale P2P IPTV System", IEEE Transactions on Multimedia, December 2007

[15] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar and Y. Wang, "Substream Trading: Towards an open P2P live Streaming System", IEEE ICNP, October 2008

[16] H. Xie, R. Yang, A. Krishnamurthy, Y. Liu and A. Silberschatz, "P4P: Provider Portal for Applications", ACM CCR, October 2008

[17] OMNet++ Simulator (http://www.omnetpp.org)

[18] Z. Li, D. H. K. Tsang and W. C. Lee, "Understanding Sub-stream Scheduling in P2P Hybrid Live Streaming Systems", IEEE INFOCOM, March 2010