

# An Alliance based Peering Scheme for Peer-to-Peer Live Media Streaming

Darshan Purandare  
pdarshan@cs.ucf.edu

Ratan Guha  
guha@cs.ucf.edu

School of Electrical Engineering and Computer Science  
University of Central Florida  
Orlando, FL 32816 USA

## ABSTRACT

Peer-to-Peer (P2P) streaming has emerged as a scalable method for media distribution in recent years. While recent measurement studies have shown the effectiveness of P2P network in media streaming, there have been questions raised about the Quality of Service (QoS), reliability of streaming services and sub optimal uplink utilization in particular. We present a new model for P2P media streaming where nodes cluster into groups, called *alliances*, for a symbiotic association in order to share the media content. We show that alliance formation is an effective way to organize the peers in loosely coupled groups. The node topology formed using alliances generates a Small World Network, which exhibit efficient overlay structures in terms of path lengths between the nodes and robustness to network perturbations like churn. We present a comparative performance evaluation of our model with CoolStreaming/DONet on QoS metrics.

## Categories and Subject Descriptors

C.2.1 [Computer-Communication Networks]: Network Operations—*Network Management*; C.4 [Performance of Systems]: Reliability, availability, and serviceability

## General Terms

Design, Performance, Reliability

## Keywords

Peer-to-Peer, Media Streaming, Quality of Service, Small World Network, Alliance, Video on Demand

## 1. INTRODUCTION

P2P streaming models like CoolStreaming/DONet (CS) [1] based on chunk driven and loosely coupled peering philosophy have become popular. Many derivatives such as PPLive [2] and SopCast [3] have evolved out of it and are

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

P2P-TV'07, August 31, 2007, Kyoto, Japan.

Copyright 2007 ACM 978-1-59593-789-6/07/0008 ...\$5.00.

largely used. Recent measurement studies have found some shortcomings of aforementioned models. A study [4] on PPLive showed that startup time of video before playback is in order of tens of seconds and sometimes even minutes and needs to be minimized for a better viewing experience. Some nodes lag in their playback time by minutes as compared to their peers. Another measurement study [5] on PPLive and SopCast revealed that these streaming models lack *tit-for-tat* fairness that leads to uneven distribution of uplink bandwidth among users. These models use greedy algorithms for peering without the consideration of peer locality that leads to huge cross ISP traffic. Moreover, due to random data distribution structures, the uplink bandwidth utilization is sub optimal. We believe that some of these problems could be alleviated by a better peering strategy.

These limitations serve as a motivation for our current work. We counter these issues by proposing a P2P model for live media streaming based on a peering scheme in which nodes cluster in groups, called *alliances*, for mutual node benefit and sharing the content. The node topology of our model forms a Small World Network [6], which is shown to be robust against network perturbations and has small overlay hops between the nodes. Using simulations, we provide a comparative performance evaluation and an empirical analysis under varying workloads and conditions of our model with CS [1], briefly described in section 2. We chose CS for comparison as it is based on swarming technology, uses chunk-driven philosophy and can serve as a benchmark. We call our model as BEAM (Bit strEAMing) [7], described in section 3. We present the details of our simulation setup and experiments in section 4. In section 5, we discuss our results. Section 6 presents the challenges and future research directions and section 7 concludes the paper with an overall discussion and summary.

## 2. RELATED WORK

CS [1] was one of the important work towards chunk driven P2P streaming. It is based on a data driven overlay network where nodes periodically exchange data availability information with a set of partners, and retrieves the unavailable data and helps peers with the deficient content. Other commercial services such as PPLive [2], SopCast [3] etc. follow a similar paradigm but their internal policies are not known. PRIME [8] is a mesh based approach that finds the global content delivery pattern to maximize the uplink utilization while maintaining the delivered quality. Redcarpet [9] focuses on providing near Video-on-Demand (VoD) and *play as you download experience* using different piece

selection algorithms in BitTorrent (BT). Since BT has been proven to be near optimal in achieving uplink utilization and mean download time, many approaches have modified BT protocol to suit the VoD needs. In this paper, we attempt to provide an efficient P2P media streaming model based on chunk driven philosophy and a unique peering scheme to counter the shortcomings mentioned in [4, 5].

### 3. OUR PROPOSED MODEL

BEAM consists of three main entities: nodes, a media relaying server and a tracker. Media relaying server is the origin of the stream content in the swarm. The tracker is a server that assists nodes in the swarm to communicate with other peers<sup>1</sup>. It also communicates with the media relaying server to exchange important information about the current state of the system. As a new user arrives in the swarm to obtain streaming services, it contacts the tracker and submits its ID, IP address and the range in which its uplink/downlink bandwidth lies. The tracker issues it a peer list, typically 40 nodes, from the set of nodes that are in similar bandwidth range. Alternatively, i.e. if it is not available, tracker provides the list of nodes in the closest bandwidth range. Interaction of nodes in similar bandwidth range leads to optimal resource utilization in the swarm [10]. The new node requests stream content from the nodes in its peer list, and starts creating and joining alliances. Alliance formation is explained in detail in Section 3.1.

The media server streams the content to a selected number of peers, termed as *power nodes*, which have higher contribution to the swarm in terms of the content served. Initially, when the streaming starts, power nodes are chosen from the nodes with higher uplink capacity, since the contribution of the nodes is yet undetermined. The power nodes in turn forward the content to other peers in the swarm. The rationale behind choosing the power nodes for direct streaming from media server is that it reduces the chances of a bottleneck at the origin of the stream content and thus prevents wastage of media server's uplink bandwidth, one of the most precious resources in the swarm.

The tracker periodically (e.g. every 10 minutes) computes the rank of the nodes in terms of the content served to the swarm. If the media server can simultaneously stream the content to, say  $P$  nodes, then the  $P$  top ranked nodes become the power nodes. The tracker updates the media server about the new power nodes, which are then streamed the media content directly from the server. The rank is calculated on the basis of a *Utility Factor (UF)*, which is a measure of the contribution of the node to the swarm.  $UF$  is computed using two parameters: *Cumulative Share Ratio (CSR)* and *Temporal Share Ratio (TSR)*. Share Ratio ( $SR$ ) is the ratio of the uploaded volume content to the downloaded volume content by an end user.  $CSR$  is the share ratio of a node since its arrival in the swarm, whereas  $TSR$  is the share ratio over a recent period of time. Thus,  $UF = f(TSR, CSR)$ . We formulate  $UF$  as follows:

$$UF = \alpha CSR + (1 - \alpha) TSR$$

where  $\alpha$  is the weight of  $CSR$  and  $(1 - \alpha)$  is the weight of  $TSR$  in calculating  $UF$ . For example, if a node has a  $CSR = 2.0$ ,  $TSR = 4.0$  and  $\alpha = 0.75$ , then  $UF = 2.5$ . Only the nodes that have  $(CSR, TSR)$  values  $\geq$  a set threshold,

<sup>1</sup>Nodes and peers have been used interchangeably.

periodically update the tracker with their  $(CSR, TSR)$ , i.e., only nodes with higher contribution report their values to tracker. This alleviates the tracker from receiving an overwhelming number of messages from all the nodes in the system. We assume that nodes are not malicious, report data honestly and do not tamper with the data, protocol and the software at the client end. Similar concept of reporting data to BitTorrent server through client software is incorporated in clients like Azureus [11].

Since the power nodes are periodically computed based on their  $UF$ , they need to upload consistently well to remain as power nodes, else they could be replaced by other well performing nodes in near future. The purpose is two fold: 1) It serves as a natural incentive for both power nodes and non power nodes to contribute to the swarm since this reward helps them to get the content early and directly from the server; the most reliable source in the swarm. Such altruism has been shown to be very effective in improving the overall swarm performance [12]. 2) Nodes with higher uploading capacity are closer to the server. Placing peers with higher uploading capacity closer to the source achieves optimal performance in terms of maximizing uplink utilization and minimizing average delay for all the peers in the swarm [10].

Nodes cluster into small groups, typically between 4 to 8, called *alliances*, to form a symbiotic association with other peers. Members of an alliance are assumed to be mutually trusted and help each other with sharing media content. Each alliance can have up to  $h$  members, and a node can be a part of up to  $k$  alliances. The parameters  $(h, k)$  are controlled to form a good connectivity through multiple paths for receiving the media content as well as to control from having an excessive number of open TCP connections, that can affect the performance [13].

#### 3.1 Alliance Formation

A node creates an alliance by sending an alliance join request to the nodes in its peer list. The receiving node can accept the alliance join request or reject it (depending on how many alliances it is currently a member of i.e.  $k$ ). In case of rejection or no reply, node times out after a short time interval and continues to search for peers to join their alliances. If a node accepts the alliance request, it issues a success message back to the requesting node. These two members of the alliance can expand and grow the alliance further. The format of a request message for an alliance is shown by:  $[A_{ID}, Num, N_1, N_2, \dots]$ , where  $A_{ID}$  is the ID of the alliance,  $Num$  denotes number of current members in the alliance, and  $N_{id}$  is the ID of the present member(s) in the alliance.  $N_1$  is the sender of the alliance join request. The format of a success message is as follows:  $[A_{ID}, Self_{ID}]$ , where  $Self_{ID}$  is the ID of the node that sends the success message to all the alliance members in  $A_{ID}$ . Nodes expand the alliance till  $k$  is reached.

#### 3.2 Alliance Functionality

A packet<sup>2</sup> comprises of  $(h - 1)$  pieces. As a member procures a new packet, it propagates the content within its alliances. It serves distinct pieces of a packet to its peers. Peers then exchange the missing pieces among themselves. This is done to leverage the uplink bandwidth of all the

<sup>2</sup>A packet refers to a collection pieces here and does not refer to an IP packet. A piece is the smallest data unit exchanged.

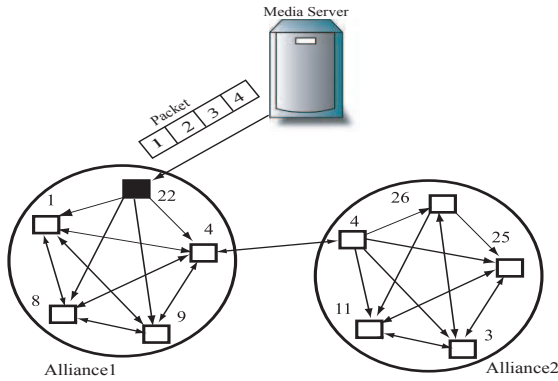


Figure 1: Alliance Functionality

peers and make participation necessary. Nodes that only download the content and do not upload to its peers are ignored by alliance members for future transactions. Alliance members replace such leecher nodes by finding another node. Consider the scenario in Figure 1, Alliance1 consists of nodes with IDs (1, 4, 8, 9, 22) and Alliance2 has nodes with IDs: (3, 4, 11, 25, 26) with node 4 being a member of both the alliances. Suppose, node 22 obtains a new packet from one of its other alliances or from media server, it then forwards it in Alliance1. It sends an announce packet to its members as:  $[A_{ID}, PNum, NPieces]$ , where  $PNum$  is the packet number in the streaming and  $NPieces$  is the number of pieces in the packet. Nodes (1, 4, 8, 9) request for unavailable pieces by sending a request in the form:  $[A_{ID}, PNum, P_1, P_2, \dots]$ , where  $P_1$  and  $P_2$  are piece number 1 and 2 respectively. Node 22 distributes distinct pieces to all members, which in turn exchange among themselves. As nodes of Alliance1 procure the complete packet, they forward it in their other alliances. In the above case, node 4 (common node in both the alliances) forwards the content in Alliance2 by announcing the arrival of the packet and the subsequent process of forwarding the content is similar as explained above.

### 3.3 Small World Network

We show that the network topology of our model which when converted to a graph, end users as vertices and connection between them as edges, forms a Small World Network (SWN) [6]. SWN is a class of random graphs where: 1) Every node has a dense local clustering and it shares an edge with some far located nodes in the graph. 2) Every node can be reached from every other node in a small number of hops or steps. We chose to show an analogy with SWN for the following reasons: 1) Overlay hops (path length) between any two nodes is short in SWN and partially reflects end to end latency [1]. 2) High local clustering means a close knit group; in a media streaming scenario it ensures that once a packet is in the alliance, it can be readily obtained from the alliance members. The important group policies required in an alliance can also be readily applied. 3) SWN are robust to network perturbations like churn and hence provide an efficient overlay structure in such events.

Nodes in an alliance forms a clique (complete subgraph), i.e. every node is connected to its alliance members (dense clustering), and shares an edge with peers in its other alliances in the swarm (other parts of the network). For e.g.,

Figure 1 depicts the neighborhood of node 4. It is a member of two alliances and in each alliance, it is connected to four other members. Node 4 is completely connected to members of Alliance1 and Alliance2, though, members of Alliance1 and Alliance2 may or may not be connected with each other. With respect to Alliance1, node 4 forms four links in the other parts of the network. Similarly, with respect to Alliance2, node 4 forms four links in the other parts of the network. This property is analogous to small-world network, where nodes are well connected locally and also have some links elsewhere in the network that helps to achieve a small path length between all pairs of nodes. Detailed description of BEAM forming a SWN and related graph theoretic properties are mentioned in [7].

## 4. SIMULATION SETUP

We present and compare the behavior of BEAM with CS under varying workloads and conditions. We quantify QoS (jitters and latency), uplink utilization and fairness (in terms of content served by an end user). The unavailability of a packet at its playback time causes a jitter. Latency is the time difference in the media playback time at server and the user end. We also analyze the robustness by evaluating the QoS of the system under sudden and gradual node failures, churn etc.

We simulate both the models by a packet level simulator i.e. BEAM as well as CS (based on the details mentioned in [1]) and compare their results based on the QoS metrics. We used the BRITE universal topology generator [14] in the Top-Down Hierarchical mode to model the physical network topology of Autonomous Systems (AS) and the routers. All AS are assumed to be in the Transit-Stub manner. Overlay is assumed to be undirected. We model both the bottlenecks in the internet, i.e. source and destination bottleneck (i.e. first-mile and last-mile hops) as the bottleneck in the non access links that are in the interior of the network, in particular within or between carrier ISP networks. The delay on inter-transit domains and intra-transit domains are assumed to be 100 ms and 50 ms respectively, while delay on stub-transit is assumed to be 30 ms and intra-stub transit links are randomly chosen between 5ms and 25ms. We model a flash crowd scenario for the arrival of users in the swarm, i.e. all users are present in the swarm when the live media streaming starts, as this is the most relevant and challenging scenario for the P2P streaming system. The uplink speed of the media server is 1536 Kb for all the experiments. The nodes in the swarm are assumed to be of heterogeneous bandwidth classes namely: (512Kb, 128Kb), (768Kb, 256Kb), (1024Kb, 512Kb), (1536Kb, 768Kb), (2048Kb, 1024Kb) where first and second member of the tuple are the maximum downlink and uplink speed of a node respectively. The distribution of these bandwidth classes is uniform in the swarm.

In our experiment, the number of nodes typically vary from 128 to 4096 for most cases. We consider a media file of duration 120 minutes, originating from a source, encoded with streaming rate of 512 Kbps and a file size of approximately 440 MB. In BEAM, we use the values of  $(h, k) = (4, 2)$  to make the neighbor count = 6, similar to CS, for a fair comparison. Table 1 provides other values of  $(h, k)$  along with the QoS obtained from those  $(h, k)$  values that can be considered for streaming. The value of  $\alpha$  is 0.75. Table 2 provides other values of  $\alpha$  that can also be considered for streaming. Each piece size is 64 Kb and hence the

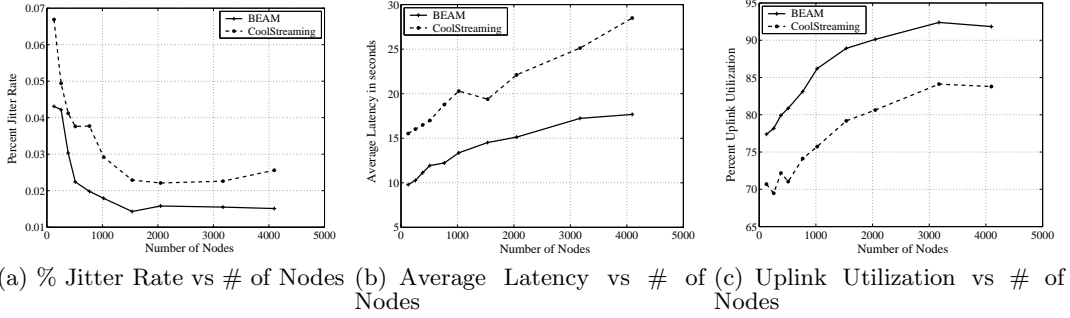


Figure 2: Comparison of QoS in BEAM and CoolStreaming

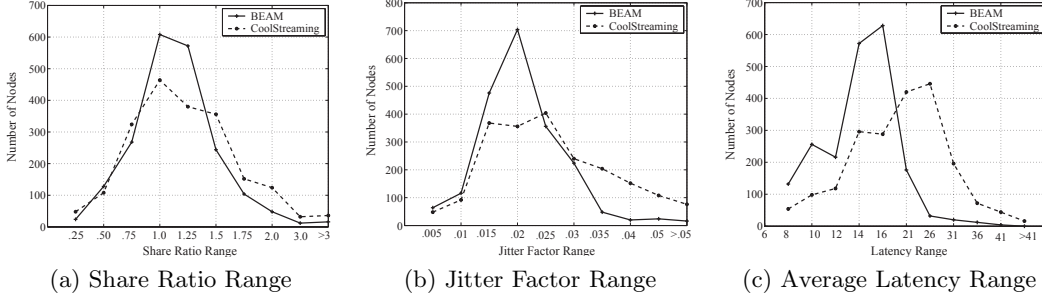


Figure 3: Number of Nodes vs Share Ratio, Jitter Factor and Latency Range in BEAM and CoolStreaming.

packet size is  $(h - 1) * 64 \text{ Kb} = 192 \text{ Kb}$  in our case. The value of  $TSR$  is set to 1 minute. The threshold share ratio is set to 2.0 for a node to report the  $(CSR, TSR)$  values to tracker. Any changes in the configuration settings are mentioned at respective sections.

## 5. RESULTS AND DISCUSSION

### 5.1 QoS and Uplink Utilization

In the first set of experiments, we compare the effectiveness of alliance theory of BEAM on QoS and uplink utilization as against CS’s random peer selection. Figure 2 depict these comparisons. From Figure 2(a), both BEAM and CS perform better with the increasing swarm size, though jitter rate slightly increases after 1500 nodes but stabilizes around 2000 nodes. BEAM has a comparatively lower (approximately 0.01%) jitter rate than CS. The reason is that in CS, the content delivery is random in nature rather than an organized flow. Sometimes an intermediate piece which could not be fetched, may increase the jitter rate. Due to alliance formation in BEAM, the stream content propagates in an organized fashion from one alliance to other; so chances of an intermediate piece missing are comparatively low. From Figure 2(b), the difference of 11 sec in average latency of BEAM and CS (for 4096 node swarm) is mainly due to the systematic flow of content from one alliance to another and near optimal overlay hops. Moreover, if a packet has been procured by an alliance member, it implies that there are at least one or more sources for the content. This flow of packets indeed saves time as compared to CS. In our implementation of both BEAM and CS, the playback starts after 6 seconds, as 6 seconds of buffer time is long enough and is many times larger than the round trip time (RTT)

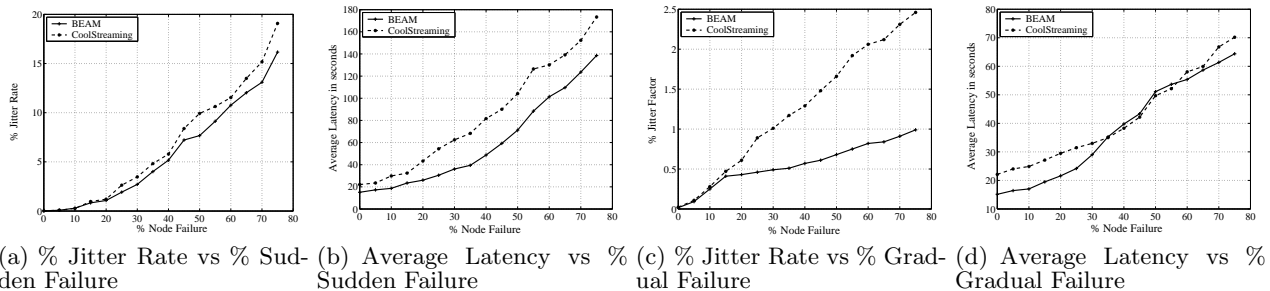
between peers to counter the network anomalies like jitter and congestion within the network [15].

From Figure 2(c), uplink utilization increases with the swarm size in both of the systems. BEAM has approximately 7% higher utilization and this is due to the fact that nodes with higher uploading capacities can effectively use their outgoing bandwidth in their other alliances, while the same may not be true for CS where a node with high uplink capacity may remain under utilized due to insufficient requests from its neighbors. In random peering (CS), neighboring peers may or may not request for pieces in the packet, while in an alliance (BEAM), members share pieces in every packet among themselves, ensuring that there are requests for uploads almost all the time. This increases BEAM’s uplink throughput. An optimal utilization of 1.0 is near impossible because of node heterogeneity and lack of download requests to the low capacity peers.

### 5.2 Fairness

End users resort to methods such as free riding, white-washing etc. in order to save their uplink bandwidth. As a result, many nodes upload much more than what they should while others get a free ride. In this paper, we quantify fairness in terms of the content served by each node (uplink bandwidth) or equivalently by their share ratios. We compute the share ratios of all the individual nodes and analyze the correlation, if any, in the QoS perceived by the nodes. Also, we study the fairness of BEAM and CS towards distributing the load evenly among users.

In Figure 3(a), we depict the share ratios of nodes and their distribution in BEAM and CS. An ideal share ratio of 1.0 is not possible in such P2P systems due to the node bandwidth heterogeneity [4, 5]. In such cases, the range of



**Figure 4:** (a), (b) depict QoS under sudden node failure, (c), (d) depict QoS under gradual node failure in BEAM and CoolStreaming.

share ratios from 0.75 to 1.25 becomes more significant since it is closest to 1.0. Larger the number of nodes having share ratio close to 1.0, fairer is the system. In BEAM, around 1180 nodes out of 2048 have their share ratios in the range 0.75 to 1.25, which forms 57.61% of the total nodes, while the distribution is more spread out in CS with 41.21% nodes lying in the region of share ratios between 0.75 to 1.25. However, some disparity can be seen in Figure 3(a) where some nodes upload more than 3 copies while others share less than one fourth of the entire content. This is because there are very few requests made to the low capacity peers, and power nodes distribute multiple copies of the content in the swarm. In CS, there are more nodes with higher share ratios and comparatively lesser nodes with share ratio closer to 1. This is attributed to the fact that some nodes with high bandwidth always remain forwarding nodes, i.e. they upload much more than the lower bandwidth nodes either because of excess bandwidth or the topology of the node in which flow could be top-down. From Figures 3(b),3(c), most nodes in the swarm receive average values of QoS parameters for both the systems.

### 5.3 Robustness and Reliability

We conducted two types of experiment to evaluate the robustness and reliability of both systems: 1) We injected various percent of node failures after 50% of the simulation run time. 2) We injected one third of node failures at three different intervals: 25%, 50% and 75% of the simulation run. We study the impact of node failures on QoS metrics and the overall system performance. This simulation run comprises of 2048 nodes. From figures 4(a) and 4(b), when the node failure is sudden, the jitter rate and latency is considerably high for both the systems as compared to when the node failure is gradual (figures 4(c), 4(d)). In the event of node failure, alliances become sparse and nodes need to find new peers to find alternate channels for content. Nodes that cannot procure stream content issue multiple requests to the nodes that have the desired content. In case of complete alliance failure, the nodes need to re-form an alliance, thereby increasing jitter and hence latency. The difference in the results of BEAM and CS can be understood in the light of SWN which displays robust behavior during churn and node failures, and has stronger edge (graph) connectivity than CS. Moreover, systematic peering in alliances show an improved performance as compared to random peering in CS.

**Table 1:** A comparison of QoS for various  $h, k$  values for a 2048 node swarm, media encoded with 512 Kbps.  $N$  denotes the number of neighbors,  $J_B/J_{CS}$ ,  $L_B/L_{CS}$  and  $U_B/U_{CS}$  denote Average Jitter Rate, Average Latency and Uplink Utilization for BEAM and CS. Values of BEAM and CS in column 3, 4 and 5 are separated by a / sign.

$h, k$	$N$	$J_B/J_{CS}$	$L_B/L_{CS}$	$U_B/U_{CS}$
$h, k = 4, 2$	6	0.0158/0.0221	15.12/22.11	90.13/80.64
$h, k = 5, 2$	8	0.0156/0.0213	15.89/21.89	91.26/82.76
$h, k = 4, 3$	9	0.0162/0.0202	16.23/20.27	92.42/84.34
$h, k = 6, 2$	10	0.0164/0.0206	17.63/22.18	90.57/85.10
$h, k = 4, 4$	12	0.0164/0.0210	16.11/23.34	88.26/84.14
$h, k = 5, 3$	12	0.0159/0.0210	15.04/23.34	92.53/84.14
$h, k = 4, 5$	15	0.0176/0.0231	17.72/23.42	86.47/83.59
$h, k = 6, 3$	15	0.0177/0.0231	17.14/23.42	89.41/83.59
$h, k = 5, 4$	16	0.0186/0.0245	17.98/24.03	85.53/80.68
$h, k = 4, 6$	18	0.0181/0.0244	18.31/24.16	86.77/82.71
$h, k = 5, 5$	20	0.0190/0.0249	18.68/26.98	87.63/84.93

### 5.4 Effect of $(h, k)$ on System Performance

Table 1 presents the performance of BEAM and CS for various values of  $(h, k)$ . Recall that  $(h, k)$  signifies neighbor count. The intent here is to find a peering scheme in terms of  $(h, k)$  such that the system performance is optimal for the QoS parameters. The performance is near optimal for most combinations of  $(h, k)$  where neighbor count is  $\leq 12$ . With increasing neighbor count, the communication overhead has an adverse effect on the QoS parameters. When the neighbor count is higher, a node that has new stream content receives more requests from its alliance members. With its limited uplink capacity, it cannot cater to all the request simultaneously, thus increasing the jitter and latency. As a result, many nodes cannot procure the content and their outgoing bandwidth is under utilized. The other values of  $(h, k)$  that we found to be near optimal are (5, 2), (4, 3) and (5, 3).

### 5.5 Effect of Power Nodes

Table 2 shows the effect of power nodes on the whole system. The optimal choices (for best QoS) were found to be  $\alpha = 0.67$  and  $\alpha = 0.75$ , though other choices were also good with marginal overhead in jitter rate, latency or uplink utilization. This may lead to a very important question: Is it necessary to change the power nodes at all during the streaming session? The answer could depend on many fac-

**Table 2: Evaluation of power nodes and their effect on the QoS factors for variable  $\alpha$ , which is the weight of CSR for calculating the UF.  $P$  denotes number of distinct power nodes during the streaming session.  $J$  denotes the Average Jitter Factor,  $L$  denotes the Average Latency in seconds and  $UU$  denotes % Uplink Utilization.**

$\alpha$	$P$	$J$	$L$	$UU$
0.00	76	0.0175	17.12	90.18
0.20	73	0.0185	17.31	89.46
0.25	67	0.0186	18.23	91.42
0.33	63	0.0175	17.66	89.45
0.50	54	0.0174	17.11	88.29
0.67	48	0.0160	16.54	91.37
0.75	45	0.0158	15.12	90.13
0.80	36	0.0162	15.78	89.27
1.00	29	0.0182	17.95	86.22

tors such as: 1) What incentive do the high capacity nodes have in contributing the content altruistically? 2) What if the already chosen power nodes decrease their uploading rate in the absence of a policy where best performers in terms of uploading are chosen as power nodes? We believe that changing the power nodes brings altruism from the high capacity peers who have an interest of being served from the server. Altruism has a very important effect on the overall efficiency of the swarm and sometimes even more than *tit-for-tat* and any kind of forced fairness policies [12].

## 6. CHALLENGES AND FUTURE WORK

We enumerate some practical engineering challenges involved in the deployment of our proposed work, and possible solutions to overcome them. We also mention the related future work. 1) A significant number of nodes (mainly home users) that use P2P streaming are behind the Network Address Translators (NAT)/Firewall. This implies that these nodes primarily act as receivers and not as transmitters in P2P streaming applications. Successful TCP NAT traversal [16, 17] can overcome most of these problems. Standardization of NAT behavior [18] and support from vendors would also ease the NAT traversal. 2) Security issues such as malicious behavior by an end user tampering the software at client end or inserting garbage content are not covered in the paper. With growing popularity of P2P streaming applications, it is a necessary to identify and address the security concerns. 3) Our research is complementary to the advanced source and channel coding techniques such as layered coding, multiple description codes (MDC), fountain codes, and network coding. Such techniques can be suitably adopted in our work.

## 7. SUMMARY AND CONCLUSION

We introduced a novel framework for P2P media streaming, *BEAM*, that uses alliance based peering scheme to solve some of the existing problems in chunk based P2P media streaming. In particular, our main contributions and findings are: 1) Peer lag (while media playback) can be significantly reduced from the order of minutes to approximately 10-20 seconds in the swarm. In *BEAM*, the initial buffering time can be reduced from approximately 30 seconds to 10-12 seconds for smaller swarms and less than 20 seconds for larger swarms (4096 nodes). Further reduction in such

buffering time to a few seconds is extremely difficult because: a) Buffering time requires the time to find the path, stream content and possible forwarders of the stream content. b) Lack of any dedicated proxy during the initial (buffering) period. c) Heterogeneity of node bandwidths in the swarm. 2) *BEAM* has displayed robust and scalable behavior while delivering near optimal levels of QoS under varying workloads and conditions. Uplink utilization has improved considerably over CS and throughput is more than 90% for larger swarms. Alliance based peering scheme encourages nodes to contribute in order to receive the content, and indeed generates a fairer swarm. Finally, we believe that our results are promising and could provide research insight towards development of newer and efficient peering strategies in P2P media streaming systems.

## 8. ACKNOWLEDGEMENTS

This work was partially supported by ARO under grant DAAD19-01-1-0502. The view and conclusion herein are those of authors and do not represent the official policies of the funding agency or the University of Central Florida, Orlando. Special thanks to anonymous reviewers whose suggestions and remarks have been very important for the paper.

## 9. REFERENCES

- [1] X. Zhang, J. Liu, Bo Li and T-S Peter Yum, 'CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming', In Proceedings of *IEEE Infocom*, March 2005.
- [2] PPLive. <http://www.pplive.com>
- [3] SopCast. <http://www.sopcast.org/>
- [4] X. Hei, C. Liang, J. Liang, Y. Liu and K.W. Ross, 'A Measurement Study of a Large-Scale P2P IPTV System', In Workshop on *Internet Protocol TV (IPTV) services over World Wide Web WWW2006*, May 2006.
- [5] S. Ali, A. Mathur and H. Zhang, 'Measurement of Commercial Peer-to-Peer Live Video Streaming', In Workshop on *Recent Advances in Peer-to-Peer Streaming*, August, 2006.
- [6] D. Watts and D. Strogatz, 'Collective dynamics of Small World Network', *Nature*, 440-442, 1998.
- [7] D. Purandare and R. Guha, 'BEAM: A Peer-to-Peer Framework for Live Media Streaming', *Technical Report CS-TR-07-04*, University of Central Florida, Orlando.
- [8] N. Magharei and R. Rejaie, 'PRIME: Peer-to-Peer Receiver-driven Mesh-based Streaming', In Proceedings of *IEEE Infocom*, 2007.
- [9] S. Annapureddy, C. Gkantsidis, P. Rodriguez and L. Massoulie, 'Providing Video-on-Demand using Peer-to-Peer Networks', *Microsoft Technical Report*, MSR-TR-2005-147, 2005.
- [10] T. Small, B. Laing and B. Li, 'Scaling Laws and Tradeoffs of Peer-to-Peer Live Multimedia Streaming', IN *ACM Multimedia Systems Track*, Santa Barbara, California, 2006.
- [11] Azureus. <http://azureus.sourceforge.net/>
- [12] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy and A. Venkatramani, 'Do incentives build robustness in BitTorrent?', In *4<sup>th</sup> USENIX Symposium on Networked Systems Design and Implementation (NSDI 2007)*.
- [13] R. Morris, 'TCP behavior with many flows', In Proceedings of *ICNP*, 1997.
- [14] A. Medina, A. Lakhina, I. Matta, and J. Byers, 'BRIT: Universal Topology Generation from a User's Perspective', *BU-CS-TR-2001-003*, April 05, 2001.
- [15] J. Lee, 'PeerStreaming: A Practical Receiver-Driven Peer-to-Peer Media Streaming System', *Microsoft Technical Report*, MSR-TR-2004-101, Sept. 2004.
- [16] B. Ford, P. Srisuresh and D. Kegel, 'Peer-to-Peer Communication Across Network Address Translators', *USENIX Annual Technical Conference*, Anaheim, CA, 2005.
- [17] B. Ford, 'Structured Streams: a New Transport Abstraction', In Proceedings of *ACM SIGCOMM 2007*, Kyoto, Japan.
- [18] IETF. <http://www.ietf.org/html.charters/behave-charter.html>