An Alliance Based Peering Scheme for P2P Live Media Streaming

Darshan Purandare, *Student Member, IEEE*, and Ratan Guha, *Member, IEEE*

***Abstract*—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. P2P streaming systems are inherently less reliable because of churn, internet dynamics, node heterogeneity and randomness in the swarm. We present a new model for P2P media streaming based on clustering of peers, called *alliances*. We show that alliance formation is a loosely coupled and an effective way to organize the peers. We show that our model maps to a "small-world" network, which form efficient overlay structures and are robust to network perturbations such as churn. We present a comparative simulation based study of our model with CoolStreaming/DONet and present a quantitative performance evaluation. Simulation results are promising and show that our model scales well under varying workloads and conditions, delivers near optimal levels of QoS, and for most cases, performs at par or even better than CoolStreaming/DONet.**

***Index Terms*—Media streaming, peer-to-peer, quality of service, small world network, video on demand.**

I. Introduction

With the advent of multimedia technology and broadband surge, there has been an increasing use of Peer-to-Peer (P2P) networks. In particular file sharing, Video-on-Demand (VoD), live event webcasting and streaming of TV channels are more common now. Various paradigms for P2P streaming have been proposed in both academia and industry. However, design flaws and inefficient peering strategy in the earlier works have led to the development of newer P2P streaming models like CoolStreaming/DONet [1] and its derivatives, most of which are built on *chunk-driven* and *loosely coupled* peering philosophy.

Users in excess of tens of thousands are turning to live streaming of popular Asian TV channels through PPLive [2], SopCast [3] and others as of 2007. Though it has been shown that P2P has emerged as a successful medium for live streaming, the Quality of Service (QoS) and reliability of streaming service still needs additional improvement. Recent measurement studies have revealed some shortcomings of current day P2P streaming systems. In particular, 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. The study further states that some nodes lag in their playback time by minutes as compared to their peers. We believe that this could be alleviated by a better peering strategy. Another measurement study [5] on PPLive and SopCast has shown that these streaming services lack *tit-for-tat* fairness which leads to uneven distribution of uplink bandwidth among users. The study found that these approaches use greedy algorithms for peering without the consideration of peer locality that leads to huge cross ISP traffic. An important finding of this work is that due to random data distribution structures, the uplink bandwidth utilization is sub optimal.

These limitations serve as a motivation for our current work. We study and attempt to counter these issues by proposing a P2P model for live media streaming based on a unique alliance based peering scheme in which nodes cluster in groups, called *alliances*, for mutual node benefit and share the content. Our work mainly focuses on leveraging the randomness of swarm like environments and imposing a few management policies at the node level to reduce the real time packet contention among the nodes. The peering strategy and internal policies in our model are unique as compared to earlier works.

We show that the node topology of our model forms a 'Small-World' Network (SWN) [6], which are shown to be robust against network perturbations and have small overlay hops between the nodes. To evaluate the effectiveness of our P2P streaming model, we quantify QoS (in terms of jitter free transmission and latency), uplink bandwidth utilization, fairness (in terms of content served), robustness, reliability and scalability of the system. Using simulations, we provide a comparative performance evaluation and an empirical analysis under varying workloads and conditions of our model with CoolStreaming/DONet (CS) [1] system. We chose to compare our model with CS because it is based on swarming technology, uses chunk-driven P2P streaming philosophy and can serve as a benchmark. Secondly many derivatives have evolved out of CS and are extremely popular among audiences. Our results show that *alliance formation* is an effective way of organizing peers and distributing content in the P2P overlay networks. We show that our model has scaled well while achieving near optimal levels of QoS. We call our model as BEAM (Bit strEAMing).

Related work is presented in Section II. We describe the details of our model in Section III. We present a graph theoretic analysis of our model in Section IV. We present the details of our simulation setup, experiments and discuss our results in Section V. We conclude the paper with an overall discussion and future research directions.

Manuscript received November 14, 2006; revised April 15, 2007. This work was supported in part by the 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. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Hui Zhang.

The authors are with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816 USA (e-mail: pdarshan@cs.ucf.edu; guha@cs.ucf.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2007.907453

II. Related Work

In the last decade many paradigms have been proposed for P2P streaming. Foremost studies suggested the use of IP multicast, but due to technical and administrative issues, its deployment has been very sparse and is not considered a feasible solution. Application layer multicast (ALM) is an alternative to IP multicast, wherein multicasting functionality is implemented at the end hosts instead of network routers. Lately, chunk-driven P2P streaming systems based on loosely coupled peering philosophy have evolved and are found to be a viable solution owing to its ease of use and deployment. Moreover, it can serve a large pool of users without the need for an additional infrastructure.

In P2P ALM, most overlay network construction algorithms form a tree like node topology. This is suitable for reliable network routers, but given the dynamics of internet, churn rate, strict time and bandwidth requirements of media streaming, these algorithms have been found to be less effective and vulnerable to failures. Noted approaches like NICE [7] and ZIGZAG [8] distributively construct an overlay network of nodes and routing functionality to minimize the number of hops for content distribution. The internal nodes in the tree are responsible for forwarding the content and any failure in these nodes causes short term failures including jitters in that particular sub tree before any repair algorithm is used for recovery. End System Multicast (ESM) [9] is a mesh based tree approach to counter the problems in tree like structures. It has the capability to send multiple video streams at different qualities to counter node failures and degrading network links.

PRIME [10] is a mesh based P2P streaming approach to live media streaming that focuses on finding the global content delivery pattern to maximize the uplink utilization while maintaining the delivered quality. Recent P2P model CoolStreaming/DONet [1] is one of the most successful approaches to live media streaming. It is based on a data driven overlay network where a node periodically exchanges data availability information with a set of partners, and retrieves the unavailable data and helps peers with deficient content. New proprietary models like PPLive [2] and SOPCast [3] etc. that are similar to [1] have gained popularity for TV streaming of Asian channels. BASS [11] is another recent P2P streaming technique for VoD that uses a hybrid approach of BitTorrent (BT) [12] and a client-server model that provides the swarm with an external media server. However, load on such server increases linearly with the number of users owing to its server centric design and hence does not scale well. BiToS [13] is a BT modified approach to VoD using the P2P network. Redcarpet [14] is yet another work that concentrates on providing near VoD and *play as you download experience* using different piece selection algorithms in BT. Since BT has been proven to be near optimal in achieving uplink utilization and mean download time, these approaches have modified BT protocol to suit the VoD needs. In this paper, we provide an efficient P2P media streaming model based on chunk driven philosophy and unique peering scheme to counter the shortcomings mentioned in [4], [5].

III. BEAM

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 (*nodes and peers have been used interchangeably*). It also communicates with the media relaying server to exchange important information about the current state of the system. As a new user arrives, it contacts the tracker and submits its IP address together with its bandwidth range. The tracker issues it a peer list, typically 40 nodes, from the set of nodes that are in similar bandwidth range. Alternatively, if it is not available, tracker provides the list of nodes in the closest bandwidth range. Small *et al.* [15] and Bharambe *et al.* [16] have shown that interaction of nodes in similar bandwidth range leads to optimal resource utilization in the swarm. The new node requests stream content from the nodes in its peer list, and then starts creating and joining alliances. Alliance formation is explained in detail in Section III-A.

Since the media relaying server cannot stream the content to multiple users simultaneously due to the bottleneck in its uplink speed, it streams the content to a selected number of peers, termed as *power nodes*, which have higher contribution to the swarm in terms of content served. Initially, when the streaming starts, power nodes are chosen from the nodes with higher uplink bandwidth, since the contribution of nodes is yet undetermined. The power nodes in turn forward the content to the other peers in the swarm.

The tracker periodically (e.g., every 10 min) 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 node utility or contribution to the swarm. UF is computed using two parameters: *Cumulative Share Ratio* (CSR) and *Temporal Share Ratio* (TSR). Share ratio 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, $\mathrm{UF} = f(\mathrm{CSR}, \mathrm{TSR})$. We formulate UF as follows:

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

where α is the weight of CSR and $(1 - \alpha)$ is the weight of TSR. For example, if a node has a $\mathrm{CSR} = 2.0, \mathrm{TSR} = 4.0$ and $\alpha = 0.75$, then $\mathrm{UF} = 2.5$. In Section V-B7, Table V presents possible combinations of (CSR, TSR) values used in determining the power nodes. Only the nodes that have (CSR, TSR) values ≥ 2.0 (empirically obtained from Fig. 5(a) and explained later) periodically update the tracker with their (CSR, TSR). These account for less than 20% of the total nodes in the swarm (see Fig. 5(a)). These 20% nodes are enough to generate the required number of power nodes in a streaming session and do not incur significant overhead since the remaining 80% nodes do not report to the tracker. This alleviates the tracker from receiving an overwhelming number of messages from the nodes in the system. We assume that nodes are honest and do not tamper with the data, protocol and the software at the client end. Sim-

ilar concept of gauging the share ratio of registered users is used in popular BitTorrent clients like Azureus [17].

Since the power nodes are periodically computed based on their UF, they need to perform consistently well in terms of distributing the content to remain as power nodes, else they could be replaced by other well performing nodes. The purpose is two fold. 1) It serves as a natural incentive for the power nodes as well as the 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 [16], [18]. 2) Nodes with higher uploading capacity are closer to the server. Small *et al.* [15] has proven that 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.

Live media streaming is time and resource constrained. Nodes contend within themselves for the same media content within a short period of time. The need to playback the media and procure the future content necessitates an effective management policy. We introduce the concept of *alliance formation* to counter these problems. 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. Our model places an upper bound on two very important parameters. Maximum number of nodes in an alliance, h, and maximum number of alliances a node can join, k. A node can be a member of at most k alliances and this helps the node to form a stronger connectivity in the pool and gives an option to receive the stream content from different alliances (paths).

As a power node receives the media content from the server, it propagates the content within its alliances. While serving the content to its alliance members, a node serves different pieces of a packet to its peers. Here, a packet refers to a collection of pieces and does not refer to an IP packet. A piece is the smallest data unit exchanged. A packet contains $(h-1)$ pieces, which the power node distributes to the other $(h-1)$ members of the alliance, i.e., each node gets one piece, which it shares with other alliance members and subsequently obtains other missing pieces from other members. In this process, a node downloads $(h-1)$ pieces and uploads $(h-2)$ pieces. This is done to leverage the uplink bandwidth of all the peers and make participation necessary so that no node gets a free ride. In case a node cannot get a particular piece because it could not fetch from a peer in its alliance, it can request the power node for it. Nodes that only procure the content from alliance members and do not share are ignored by other alliance members in future and alliance member find another node to replace such non contributing node to be in the alliance. As a node gathers all the pieces of a packet, it starts the media playback, forwards the content among its other alliances like power node and procures the future stream content. A node uses *announce mechanism* to notify its alliance members the receipt of a new packet. This process of announcing and exchanging unavailable content can also be efficiently improved using Network Coding [19]. Periodically, nodes in an alliance serve a request that is out of the alliance to bootstrap a new node.

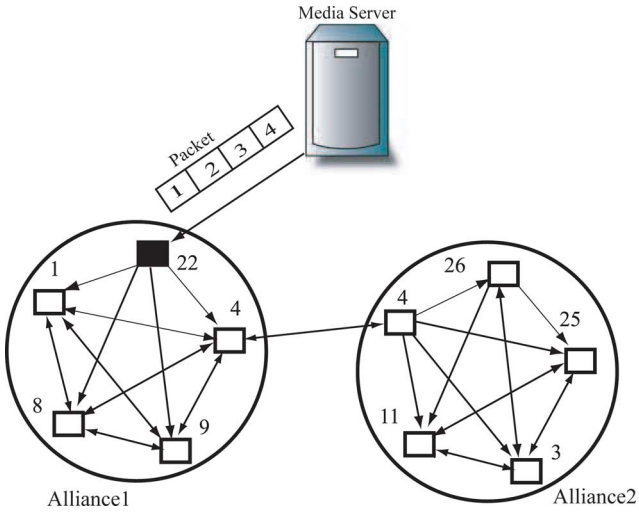

Fig. 1. Alliance functionality.

A. Alliance Formation

A node creates an alliance by sending an alliance join request packet 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 [e.g., 2 round trip times (RTT)] and continues to search for peers to join their alliances. If a node accepts the alliance request, it issues a success packet back to the requesting node. These two members of the alliance can expand and grow the alliance further. The format of a request packet for an alliance is shown by $[A_{\mathrm{ID}}, \mathrm{Num}, N_1, N_2, \ldots]$, 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 packet is as follows: $[A_{\mathrm{ID}}, \mathrm{Self}_{\mathrm{ID}}]$, where $\mathrm{Self}_{\mathrm{ID}}$ is the ID of the node that sends the success message to all the alliance members in A_{ID}.

B. Alliance Functionality

A node can be a member of multiple alliances (at most k). This is important to facilitate multiple paths for a node to obtain the stream content in case of node failures. As a member of an alliance procures a packet, it spreads it among its respective alliances. Consider the scenario in Fig. 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 $[A_{\mathrm{ID}}, P\mathrm{Num}, N\mathrm{Pieces}]$, where $P\mathrm{Num}$ is the packet number in the streaming and $N\mathrm{Pieces}$ is the number of pieces in the packet. Nodes (1, 4, 8, 9) request for unavailable pieces that they need to procure to complete the download by sending a request in the form $[A_{\mathrm{ID}}, P\mathrm{Num}, P_1, P_2, \ldots]$, where P_1 and P_2 are piece numbers 1 and 2, respectively.

A packet comprises of $(h-1)$ pieces. If all the members of a particular alliance simultaneously request all the pieces, the for-

warding node randomly distributes the pieces of the requested packet among them. It is left to the peers to exchange the pieces within themselves. In case, a node requests specific unavailable pieces in a packet since it has already obtained some pieces from other alliances, the forwarding node sends only those specific requested pieces to avoid any redundancy at the requesting node. In the above example, if members of Alliance1 have procured distinct pieces, they exchange those pieces among themselves to complete their individual downloads. As nodes of Alliance1 procure the complete packet, they forward it to 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.

While leaving the network, a node sends a departure packet to its alliance members as follows: $[A_{\mathrm{ID}}, \mathrm{Self}_{\mathrm{ID}}, \mathrm{Flag}_D]$, where Flag_D is the departure flag. In case a node exits without sending a departure packet, other nodes within the alliance become aware of its inactivity and infer its departure. Other nodes in the alliance continue sharing the streaming content within themselves and/or can find another member for the alliance. In our model, we propose to use TCP connection in BEAM as the network links between peers. TCP detects the peer's departure from the pool and gracefully handles shutdown of the connection link. Li [20] has explained the benefits of using TCP over UDP/RTP, provided the initial buffer time is significantly larger than the Round Trip Time (RTT) between the peers. We elaborate the details in the simulation section.

IV. Small World Network

In this section, we present an analogy between BEAM and Small World Network (SWN) [6] to show the effectiveness of BEAM's important properties such as near-optimal overlay distance and network robustness in the events of churn and node failures. SWN is a class of random graphs where the following is true. 1) Every node has dense local clustering, i.e., a high coefficient of clustering (μ_c, defined below) and some edges with far located nodes. 2) Each node can be reached from every other node by a small number of hops or steps. We present a graph theoretic analysis of our model and show that it generates a swarm of nodes, which when converted to a graph, end users as vertices and connection between them as edges, exhibits small world network characteristics. We compare our results with CS [1] which uses a random network topology. Random graphs [21] are known to generate near-shortest mean distance between all pairs of nodes in a graph.

We chose to show an analogy with small world network 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], [7], [8]. 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 the event of node failure.

A. Alliances and Small World Network

μ_c is a local property of a vertex v in a graph and is defined as follows. Consider the vertex v and a set of neighboring vertices $V = (v_1, v_2, \ldots, v_n)$ and a set of edges E, where e_{ij} denotes an edge between the vertex i and vertex j. The clustering coefficient(μ_c) of a vertex is the ratio of actual number of edges present to the total possible edges among those vertices

$$\mu_c = \frac{|e_{ij}|}{\binom{n}{2}} = \frac{2|e_{ij}|}{n(n-1)}.$$

In other words, μ_c is density of edges in the node's vicinity. Average of clustering coefficients of all the nodes is the clustering coefficient of the graph. Mean path length is the mean of path lengths between all the pairs of vertices in the graph. The concept of SWN is counter intuitive as graphs with higher clustering coefficients would be dense locally and require more hops to traverse the other parts of the graph as compared to a random graph. Watts *et al.* [6] showed that routing distance in a graph is small if each node has edges with its neighbors (i.e., has high μ_c) as well as some randomly chosen nodes in the swarm. Similarly, Kleinberg [22] proved that if every node in the swarm shares an edge with a far located node, the number of expected hops for routing between any pair of vertices becomes $O(\log^2 N)$.

Suppose a node is a member of k alliances $(a_1, a_2, \ldots a_k)$ and each alliance has neighbors $(m_1, m_2, \ldots, m_k)$, where $|m_i| \leq h$, and $1 \leq i \leq k$. Therefore, coefficient of clustering for such node would be

$$\mu_c \geq \frac{\binom{m_1}{2} + \binom{m_2}{2} + \cdots + \binom{m_k}{2}}{\binom{m_1 + m_2 + m_3 \ldots + m_k}{2}}.$$

Fig. 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. Nodes in an alliance forms a clique (complete subgraph). 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 long distance links elsewhere in the network. Similarly, with respect to Alliance2, node 4 forms four long distance links elsewhere in the network. This property is analogous to small-world network, where nodes are well connected locally and also have some long distance links elsewhere in the network that help to achieve a small path length between all pairs of nodes. Coefficient of clustering for node 4 in this case would be

$$\mu_c \geq \frac{\binom{4}{2} + \binom{4}{2}}{\binom{8}{2}} = \frac{3}{7} = 0.428.$$

We also consider the case where other alliance members can have edges between them. Similarly, other nodes in the graph would have μ_c of at least 0.428 since they have similar constitution of alliance and neighborhood. Clustering coefficient of 0.428 is relatively much higher than a random graph (μ_c for

random graph of the same size was found to be 0.0019), and therefore it lies in the region of small world graphs as mentioned in [6].

B. Graph Theoretic Properties of Alliance

Graph density (ratio of number of edges to the total number of possible edges in the graph) is an important factor for the connectedness of a graph. We evaluate the graph density of a BEAM graph by abstracting the alliances as nodes. As a member of an alliance receives a packet, it forwards within its alliance members and hence we focus on alliance hops rather than individual node hops in this scenario and compute the same. To simplify, we consider an alliance as a single node which we call super node. Suppose there are N nodes in the swarm viz $(V_1, V_2, \ldots, V_N)$ and they are spread in M alliances. Let D_{graph} be the density of the graph, D_{alliance} be the density of the graph when alliances are abstracted as vertices i.e., super nodes as vertices, M be the number of super nodes in the swarm, O be the outdegree of a super node. Every node in the swarm is connected to $(h-1)$ other nodes in every alliance and there are k such alliances. Therefore, we have

$$D_{\text{graph}} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{k}(h_{ij}-1)}{2*\binom{N}{2}} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{k}(h_{ij}-1)}{N*(N-1)}$$

where h_{ij} is the number of members in jth alliance of node i, and $1 \leq i \leq N, 1 \leq j \leq k$. In a steady state, when all the nodes have formed k alliances, and each alliance has exactly h members, we have

$$D_{\text{graph}} = \frac{(h-1)k}{N-1}$$

Since super nodes are formed by contracting the alliances, we have

$$M = \frac{Nk}{h}.$$

Every super node is connected to other super nodes through its h members and their respective $(k-1)$ alliances since every node is a member of k alliances each. Therefore, we have

$$O = h(k-1).$$

Since there are M super nodes and each has a outdegree of O, there are $(MO/2)$ edges. Therefore, we have

$$D_{\text{alliance}} = \frac{\frac{MO}{2}}{\binom{M}{2}} = \frac{h^2(k-1)}{(Nk-h)}.$$

For $h = 5, k = 2$, i.e., node $\text{degree} = (h-1)*k = 8$ and $N = 512$, the D_{graph} is approximately 0.004, while D_{allaince} is approximately 0.025. We see that the density of the graph at alliance level is relatively much higher than at the node level i.e., $D_{\text{alliance}} \gg D_{\text{graph}}$. The alliance formation and subsequently the topology of the network produces strongly connected graph and reduces the hop count during the communication. Similar abstraction is not possible for complete random graphs.

TABLE I
COMPARISON OF BEAM, RANDOM AND HYBRID GRAPH FOR 512 NODES WITH NODE DEGREE 8. HERE, G IS TYPE OF GRAPH, D IS DIAMETER, R IS RADIUS, M IS MEAN DISTANCE, S IS SERVER DISTANCE AND C REPRESENTS CLUSTERING COEFFICIENT

G	D	R	M	S	C
BEAM	6	5	3.37	3.19	0.42
Hybrid	5	4	3.33	2.87	0.014
Random	5	4	3.26	3.16	0.013

TABLE II
LISTS OF ALL THE MEDIA STREAMING METRICS USED IN THE SIMULATIONS. AJ IS AVERAGE JITTER FACTOR, AL IS AVERAGE LATENCY AND UU REPRESENTS THE UPLINK UTILIZATION. OTHER RELATED TERMS ARE EXPLAINED IN TABLE III

Term	Expression	Description
AJ	A_j	$\left(\sum_{i=0}^{N} J_i\right)/N$
AL	A_l	$\left(\sum_{i=0}^{N} L_i\right)/N$
UU	U	$Uplink\ Used/Uplink\ Available$

We are more interested in the mean path lengths from the server to the nodes rather than the mean path lengths between all pairs of nodes. Therefore, we limit our search to the length of all the paths from the server to all other nodes in the swarm. To gauge the actual path length in a large swarm, we conducted experiments for finding the average path length between all nodes, average path length from server, radius, and diameter of a graph.

Table I illustrates results from a simulation in which we compare synthetically generated random graphs with BEAM graph having the same node degree and overall density in the graph. We use networkx python library [23] for complex networks to obtain our simulation results. In these experiments, we have tested three kinds of graphs: completely random graph, BEAM network graph, and hybrid BEAM graph where alliances act as nodes. The graph density, node $\text{degree} = 6$, and the node $\text{count} = 512$ were the same in all the three graphs. Hybrid graph's node count is reduced to $(Nk/h) = 256$ since $h = 4, k = 2$, and the degree of such hybrid nodes is equal to h. Server distance is calculated by picking a random node among the 512 nodes and then calculating distance from it. From the results in Table I, it is seen that random graphs perform well as expected in all the metrics. BEAM graphs have performed at par with the random graphs. Server distance in hybrid graphs is shorter than random graphs. Random graphs have relatively lower mean path length while hybrid graphs have the lowest mean server distance. This abstraction of BEAM graphs helps to analyze the topography and various other graph theoretic properties.

Fig. 2 depicts the performance of random and BEAM graphs for higher number of nodes. Random graphs are known to perform better while traversing the graph. The figure shows that it has relatively shorter radius and diameter as compared to BEAM graph. BEAM has nearly matched the random graphs in mean distance from the server, which is the most important criteria in our environment. It is an indicative of the overlay hops a

TABLE III
THIS TABLE EXPLAINS ALL THE METRICS RELATED TERMS

Term	Expression	Description
Total Number of Nodes	N	Number of Nodes in swarm
Total Number of Packets	T	Total Packets in a Streaming Session
Number of Server Upstream Connection	P	Server can simultaneously upload to P nodes
Total Uploads by node i	UP_i	Volume of Uploads by node i
Total Downloads by node i	$DOWN_i$	Volume of Downloads by node i
Share Ratio of node i	SR_i	$UP_i/DOWN_i$
Media Playback Time at Server	T_{server}	Start Time of Media Playback at Server
Media Playback at Node i	T_i	Start Time of Media Playback at Node i
Piece Availability	F_i	0 if packet available before media playback else 1
Jitter Factor of Node i	J_i	$\left(\sum_{i=0}^{T} F_i\right) \Big/ T$
Latency of node i	L_i	$L_i = T_{\text{Server}} - T_{\text{Node}_i}$

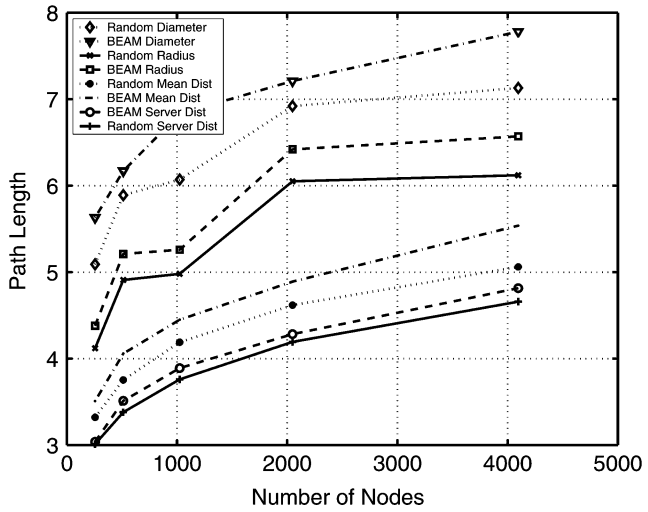

Fig. 2. Path length versus Number of Nodes for various graph parameters in Random and BEAM graphs.

node is away from the server. Thus, the BEAM graph forms a small world network with relatively high clustering coefficient and very comparable in mean path length to the random graphs.

V. SIMULATION SETUP AND EXPERIMENTS

A. Simulator Details

We simulate both the models i.e., BEAM as well as CS and compare their results based on the metrics defined in Table II. Table III explains all the related terms in it. We simulate all the components of CS as mentioned in [1]. In BEAM, we model the server and tracker functionality, and the nodes in the swarm. Server is the only source of streaming packets in the system. For comparing the two systems, we quantify QoS (in terms of jitter factor and latency), uplink utilization, fairness (in terms of content served by an end user). We also analyze robustness, reliability, and scalability of the system by evaluating the QoS of the system under varying workloads and conditions like node failure, churn, larger swarms etc.

We used the BRITE universal topology generator [24] 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. Unlike other simulators [10], [16], we assume that the bottleneck in the network can appear in the access links of source and destination (i.e., first-mile and last-mile hops) as well as the non access links that are in the interior of the network, in particular within or between carrier ISP networks. The nodes in the swarm are assumed to be of heterogeneous bandwidth classes namely: (512 Kb, 128 Kb), (768 Kb, 256 Kb), (1024 Kb, 512 Kb), (1536 Kb, 768 Kb), (2048 Kb, 1024 Kb) 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. To simulate the congestion in the Internet, we induce 5% congestion in the non access links within the interior of the network. In such congestion scenarios, the available bandwidth to nodes is the minimum of the bottleneck at source or destination and the bottleneck in the non access links. 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 5 ms and 25 ms. We simulate the TCP level dynamics like timeouts, slow start, fast recovery and fast retransmission by introducing a delay of 10 RTTs [25]. 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.

In our experiment, the number of nodes typically vary from 128 to 4096. For some large sets of experiments we have also considered nodes in excess of 16 000. We consider a media file of duration 120 min, 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 $\text{count} = 6$, similar to CS, for a fair comparison. Table IV provides other values of (h, k) that can be considered for streaming. The values of α is 0.75 from Table V. Each piece size is 64 Kb and hence packet size is $(h-1)*64\text{Kb} = 192$ Kb in our case. The value of TSR is set to 1 min. 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.

B. Results and Discussion

1) Qos and Uplink Utilization: In the first set of experiments, we compare the effectiveness of alliance theory of BEAM on

TABLE IV
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

TABLE V
EVALUATION OF POWER NODES AND THEIR EFFECT ON THE QOS FACTORS FOR VARIABLE α, 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

α	**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

QoS and uplink utilization as against CS's random peer selection. Fig. 3 depict these comparisons. In accordance with the conventional notion of scalability in P2P systems, it can be seen from Fig. 3(a) that BEAM and CS both perform better with the increasing swarm size, though jitter rate slightly increases after 1500 node mark but stabilizes around 2000 nodes. BEAM has a comparatively lower (approximately 0.01%) jitter rate than CS. The plausible 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 the other; so the chances of an intermediate piece missing are comparatively low. In BEAM, every node receives the content through the best possible channel among its various alliances, while the same cannot be commented for CS. An optimal jitter rate of 0 is difficult to attain in such random swarm environments because the content distribution is dynamic and, lasts for an extended time; network anomalies and congestions can cause unavailability of a packet at its playtime.

Fig. 3(b) depicts the average latency in both the systems. For the same settings, average latency for BEAM varies from less than 10 s for a 128 node swarm to less than 18 s for a 4096 node swarm, while CS has considerably higher latency of 29 s for a swarm of 4096 nodes. An explanation for this could be that higher the number of hops in the overlay, greater are the chances of increased end to end latency [7], [8]. CS has comparatively higher bootstrapping time before playing media and

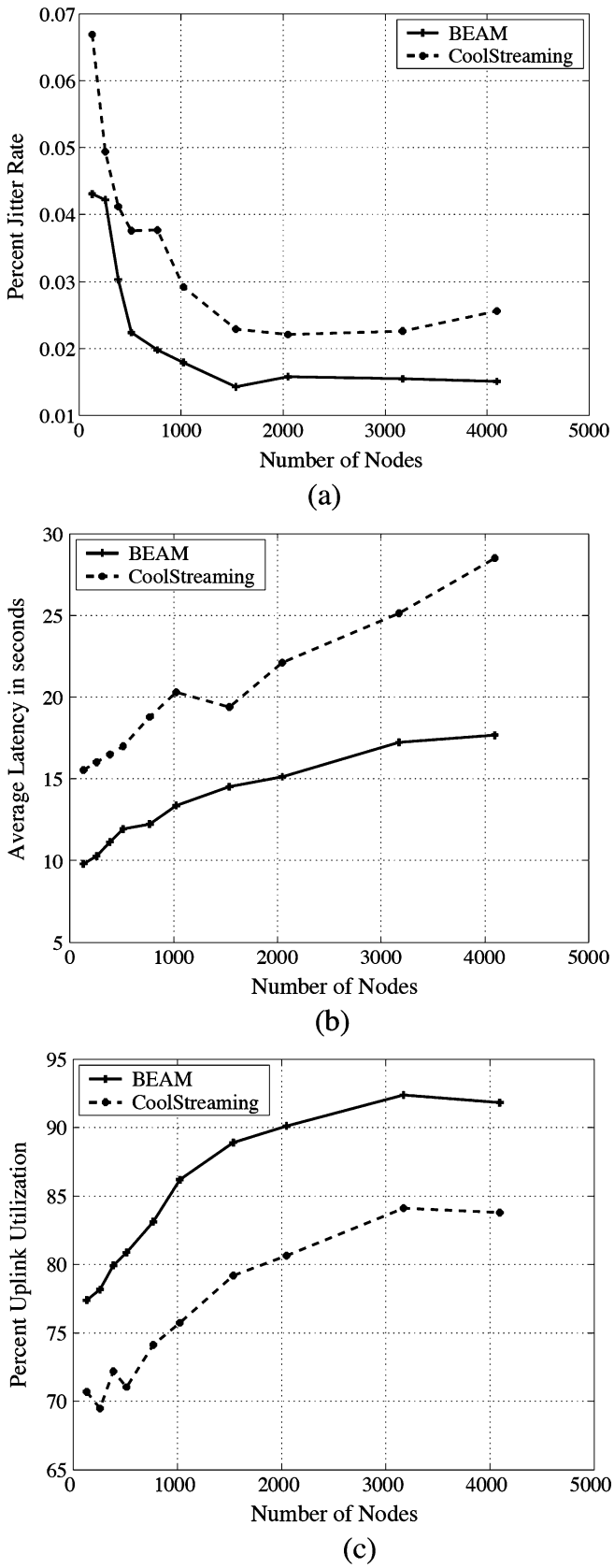

Fig. 3. Comparison of QoS in BEAM and CoolStreaming. (a) % Jitter Rate versus Number of Nodes, (b) Average Latency versus Number of Nodes, and (c) % Uplink Utilization versus Number of Nodes.

it could be attributed to the following facts. 1) It buffers more pieces in advance before playback. 2) Due to random nature of data exchange, a missing intermediate piece further increases jitter and hence latency. 3) Execution of intelligent scheduling algorithm causes both computation overhead and delay. On the contrary, in BEAM, the systematic flow of content from one alliance to another and near optimal overlay hops account for its lower latency. 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. Playback starts 10 s after receiving the first segment in [1]. In our implementation of both BEAM and CS, the playback starts after 6 s, as 6 s of buffer time is long enough and is many times larger than the RTT between peers to counter the network anomalies like jitter and congestion within the network [20].

Uplink bandwidth is the most important resource of a P2P system. End users that are charged for bandwidth used per time unit want to maximize their utilization. Moreover, maximization of uplink bandwidth is a must for a scalable system [15]. From Fig. 3(c) it is clear that uplink utilization increases with the swarm size in both of the systems. BEAM has approximately 7% higher utilization and this could be due to the fact that nodes with higher uploading capacity 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 packer, while in an alliance (BEAM), members share pieces in every packet among themselves, ensuring that there are requests for upload almost all the time, this increases BEAM's uplink utilization. An optimal utilization of 1.0 is near impossible because of node heterogeneity and lack of download requests from the low capacity peers.

2) Streaming Rate: We vary the streaming rate from 64 Kbps to 512 Kbps in a 2048 node swarm and expect a comparative deterioration in QoS with increasing streaming rate as the nodes need to procure more content for the same playback time. For lower streaming rates, QoS is expected to be near optimal as additional packets are fetched much before their playtime and chances of jitter and hence latency become negligible. From Fig. 4(a), the difference in average jitter rate between BEAM and CS is marginal for lower streaming rates but more prominent for higher streaming rates when the systems are subjected to stress test. In Fig. 4(b), similar trends can be observed for average latency while analyzing the effect of varying streaming rate on BEAM and CS. For the same playback time, a node needs to obtain higher number of packets that incurs additional time overhead. Fig. 4(c) shows the variation in uplink utilization of both the systems. They both peak around encoding rate of 256 Kbps. A plausible reason could be that at this encoding rate, the nodes are able to cater all the requests at optimum rate and this in turn increases the throughput. Node topology and peering partners are important in analyzing the utilization of bandwidth. Most commercial websites stream at rates between 225 Kbps and 450 Kbps as of 2007. Receiver should have $\text{downlink} \geq \text{streaming rate}$ and sender should have enough uplink to contribute. In our simulation, the bandwidth classes of lowest strata is 512 Kbps, so we have limited our discussion to streaming rate of 512 Kbps. 512 Kbps can be considered as

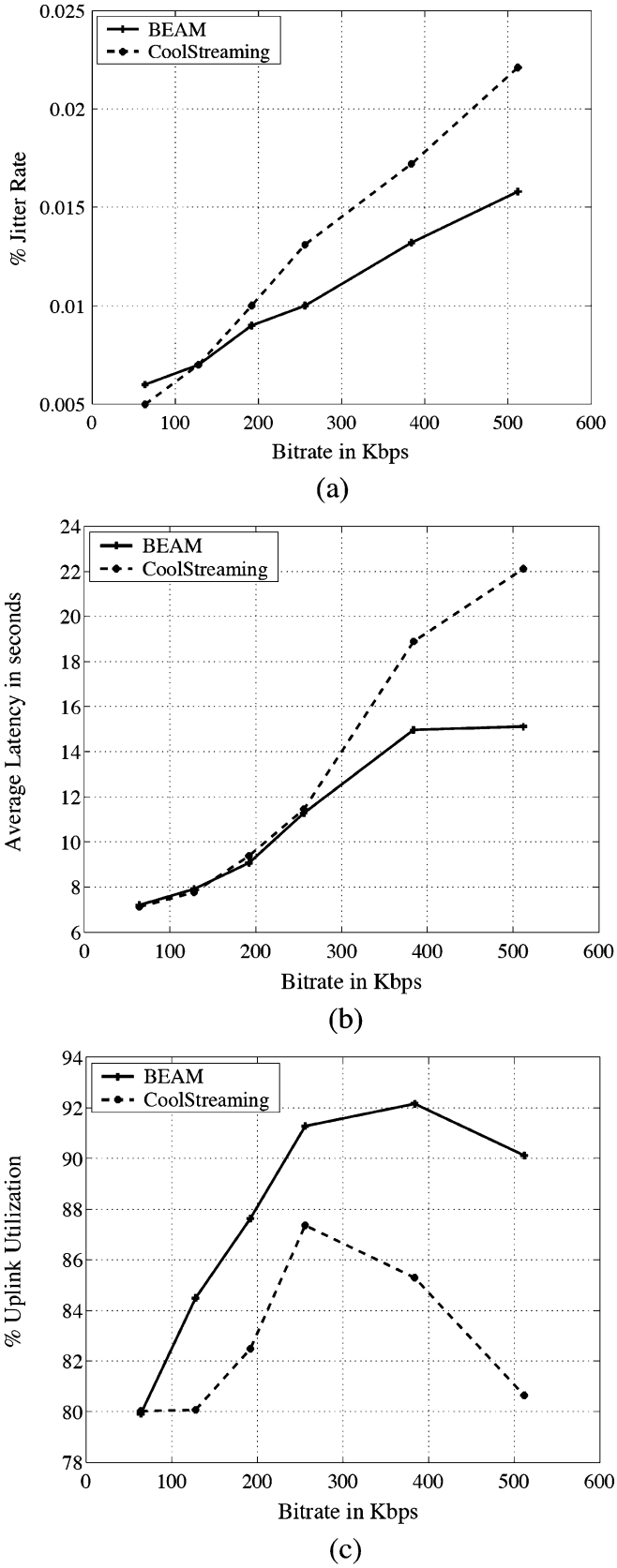

Fig. 4. Effect of Bitrate on the QoS in BEAM and CoolStreaming. (a) % Jitter Rate versus Bitrate, (b) Average Latency versus Bitrate, and (c) % Uplink Utilization versus Bitrate.

a decent rate, though in near future streaming of DVD quality media will require additional bandwidth.

3) Fairness: End users resort to methods such as free riding, whitewashing 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 Fig. 5(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 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 Fig. 5(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 Fig. 5(b), (c), most nodes in the swarm receive average values of QoS parameters for both the systems.

4) 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 Fig. 6(a) and (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 (Fig. 6(c), (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. We observe that in the case where the node departure is gradual (at specific time intervals), the node recovery is comparatively easy and makes system inherently more stable since more time is available for recovery. For example, when the node failure occurs say within the first 25% of the simulation run time, the system is recovered much before another failure occurs at 50% simulation run time. Similarly, when another failure occurs at 75% simulation time the system is already stable. The difference in the results of BEAM and CS can be understood in the light of SWN which displays robust behavior during churn

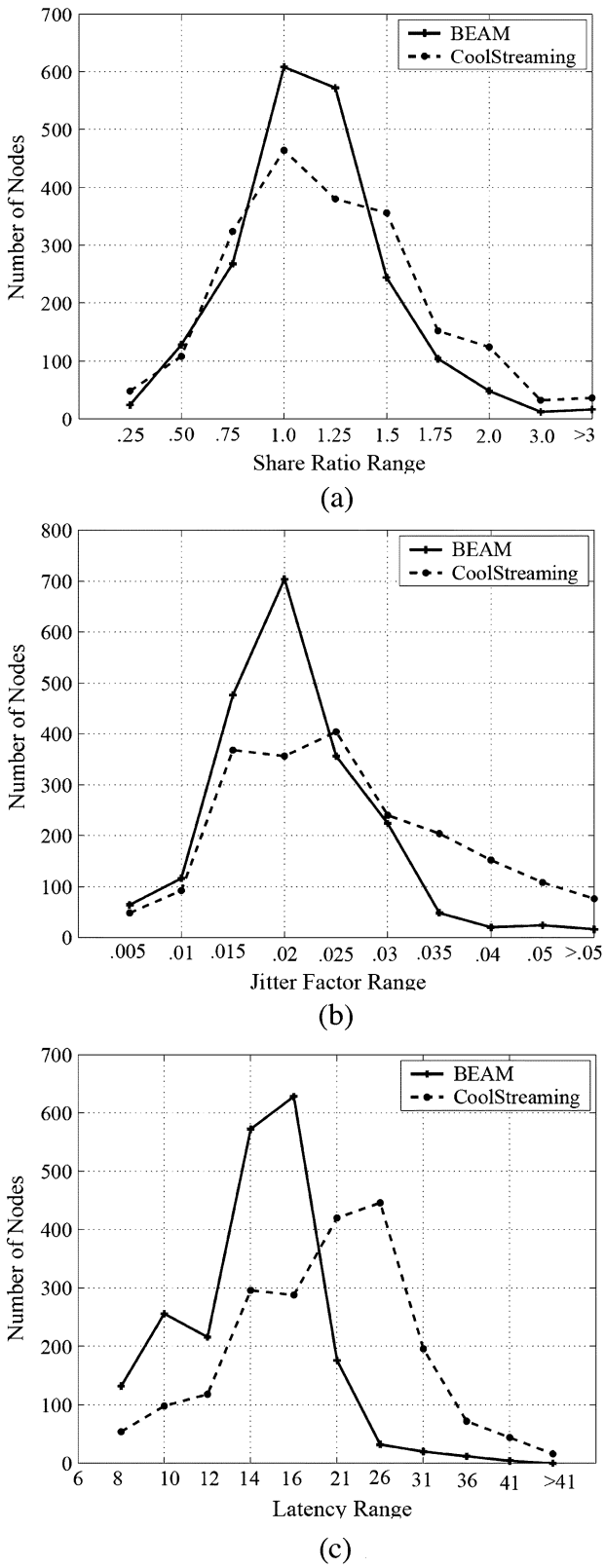

Fig. 5. Number of Nodes versus Share Ratio, Jitter Factor Range and Latency Range in BEAM and CoolStreaming. (a) Share Ratio Range, (b) Jitter Factor Range, and (c) Average Latency Range.

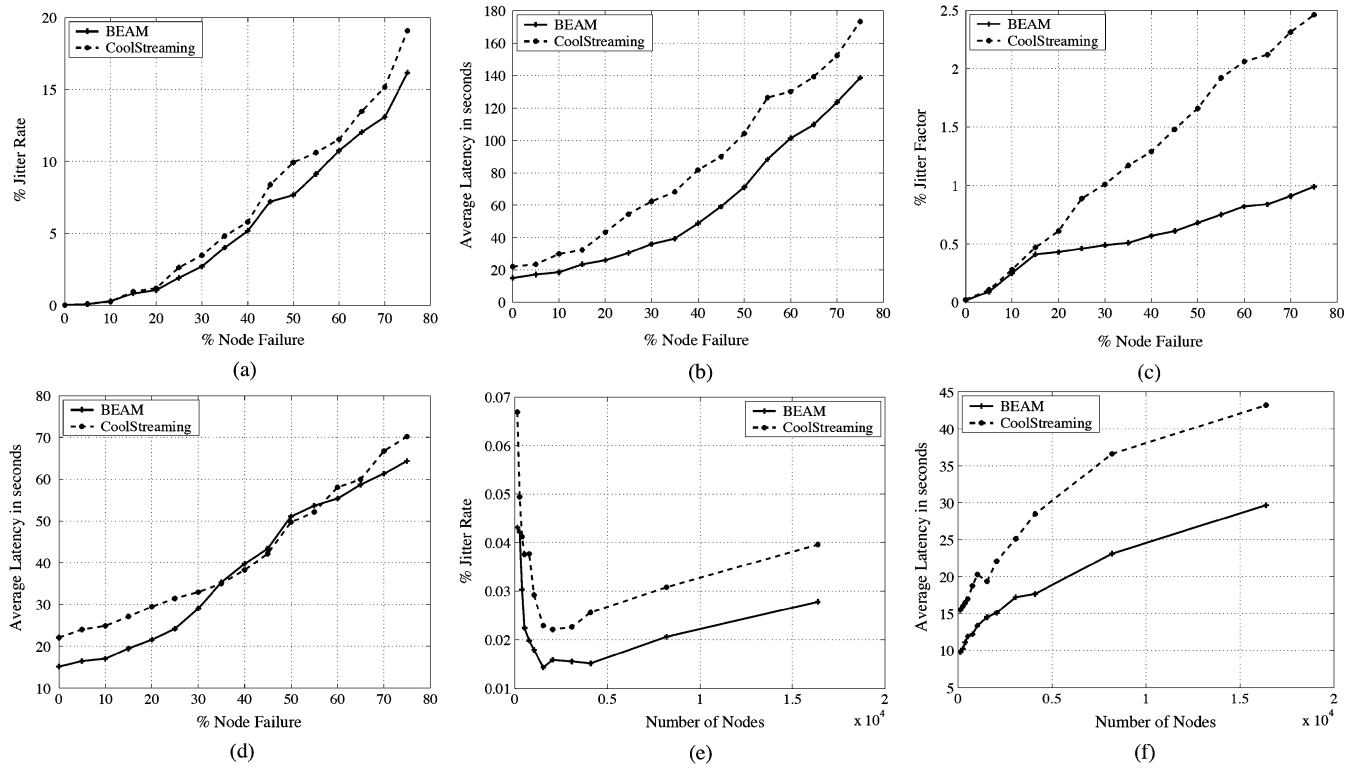

Fig. 6. (a), (b) depict QoS under sudden node failure, (c), (d) depict QoS under gradual node failure, (e), (f) represent QoS for very large swarms in BEAM and CoolStreaming. (a) % Jitter Rate versus % Sudden Failure, (b) Average Latency versus % Sudden Failure, (c) % Jitter Rate versus % Gradual Failure, (d) Average Latency versus % Gradual Failure, (e) % Jitter Rate for Larger Swarm, and (f) Average Latency for Larger Swarm.

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.

5) Scalability: In this section, we extend the results obtained for QoS and uplink utilization to larger swarms. We evaluate scalability in terms of maximum number of nodes a streaming system can support without degrading the QoS. In this experiment, we vary the number of nodes from 128 to 16384. From Fig. 6(e), we observe that for a swarm size of 16384 nodes, the average jitter rate is around 0.0278% in BEAM and almost around 0.04% for CS. With increasing nodes, jitter factor decreases and becomes steady after 1500 node count and marginally increases for very large swarms. However, even with a steep rise in the number of nodes in the swarm the average jitter factor is found to be under acceptable levels. The difference between CS and BEAM is more evident for larger swarms.

From Fig. 6(f), in BEAM, the average latency is under 30 s for a 16384 node swarm. The peer lag is approximately less than 20 s. As the number of users in the swarm increase, there are more alliances and as the content is forwarded from one alliance to other, the number of total hops increase resulting in a higher latency. CS and BEAM have a comparable performance except that CS takes a little more time to bootstrap. One of the important problems in CS like models is the high peer lag for media playback and high buffering time. BEAM has displayed considerable improvement in both aspects, i.e., reduced peer lag and reduced initial buffering time from more than half a minute to approximately 20 s.

6) Control Overhead: Control overhead is the ratio of the total number of bytes expended in communication and control

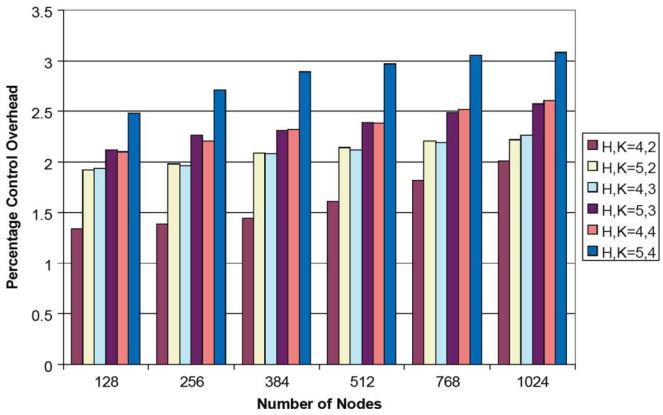

Fig. 7. Control Overhead in BEAM for various values of h and k.

to the total bytes used for streaming data payload. An efficient system aims to minimize the control overhead (CPU time and bandwidth) and maximize resource utilization towards the streaming content. Fig. 7 shows the communication overhead incurred in varying swarm sizes ranging from 128 to 1024 nodes. We vary the values of h and k. Recall, that values of h and k denote the node degree. With higher node degree, additional resources are needed resulting in an increased communication and control overhead. For 1024 node swarm and $(h, k) = (5, 4)$ $(\text{node degree} = 16)$, the control overhead is slightly over 3%. For most other permutations of h and k values, the control overhead is around 2%. Table IV shows effect of various values of h and k on the QoS parameters.

We found $(h,k) = (5,2), (h,k) = (4,3)$ and $(h,k) = (5,3)$ as well performing schemes. In our experiments in the paper, we use $h = 4$ and $k = 2$, to show a comparison with CS which has a neighbor count of 6. As mentioned in [1], for a 200 node swarm in CS, the control overhead is around 2% for node degree of 6, which is quite comparable to BEAM. BEAM and CS almost incur similar overhead. In BEAM, a node sends announce request as it receives a packet, while in CS nodes send information packets to all their neighbors periodically about their buffer state.

7) Chain Effect of Power Nodes: Chain effect due to change of power nodes in the system could disrupt many on going sessions because the focal point of data delivery from media server changes in the swarm. Table V shows the chain effect on our three main metrics, i.e., jitter, latency and uplink throughput. For various values of α (the weight factor for CSR and TSR), we compare the jitter factor, average latency, and uplink utilization. $p\mathrm{Nodes}$ is the number of distinct nodes that were chosen as power nodes at least once during the streaming session. The optimal choices 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, given that such periodic computation of power nodes may lead to a series of chain effect ?* The answer could depend on many factors. What incentive do the high capacity nodes have in contributing the content altruistically? What if the already chosen power nodes decrease their uploading rate (in the absence of such a policy where best performers in terms of uploading are chosen as power nodes)? We show 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 [18]. Moreover, a buffer time of 6 s is long enough to provide tolerance to network anomalies like chain effect, and results indeed show that BEAM can achieve near-optimal values of QoS even with periodic change of power nodes.

VI. 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 as follows. 1) Peer lag (while media playback) can be significantly reduced from the order of minutes to approximately 10–20 s in the swarm. Initial buffering time can be reduced from around 30 s to 10–12 s for smaller swarms and less than 20 s for larger swarms (4096 nodes) in BEAM. 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 then 90% for larger swarms. Alliance based peering theory encourages every node to contribute in order to receive the content, and indeed generates a fairer swarm.

A possible challenge in our work is the non standardization of Network Address Translation (NAT). A significant number of nodes (mainly home users) that use P2P media streaming are behind the NAT boxes. This implies that these nodes primarily act as receivers and not as transmitters in P2P streaming applications. Successful TCP NAT traversal can overcome most of these problems, however, in cases that include multiple levels of NAT and when both hosts are behind NAT [26], alternative strategies need to be devised. A new technique called as TCP hole punching [27] has shown to resolve this problem, though its implementation is not widespread as of now. SST [28] provides the functionality of TCP and can traverse existing NATs effectively. Standardization of NAT behavior by vendors can further facilitate the implementation [29]. Lately, NAT vendors are supplying boxes with support for TCP NAT traversal. Manually configuring home routers is also seen as an option to overcome the NAT traversal problem [17]. 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.

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.

Acknowledgment

Special thanks to anonymous reviewers whose suggestions and remarks have been very important for the paper.

References

[1] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming," in *Proc. IEEE INFOCOM 2005*, 2005, vol. 3, pp. 2102–2111.
[2] *Pplive*, [Online]. Available: http://www.pplive.com
[3] *Sopcast*, [Online]. Available: http://www.sopcast.org
[4] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "Insights into pplive: A measurement study of a large-scale p2p iptv system," in *Proc. IPTV Workshop, Int. World Wide Web Conf.*, 2006.
[5] S. Ali, A. Mathur, and H. Zhang, "Measurement of commercial peer-to-peer live video streaming," in *Proc. ICST Workshop Recent Adv. Peer-To-Peer Streaming*, Waterloo, ON, Canada, 2006.
[6] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998.
[7] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM 2002*, New York, Oct. 2002, vol. 32, no. 4, pp. 205–217.
[8] D. A. Tran, K. A. Hua, and T. T. Do, "Zigzag: An efficient peer-to-peer scheme for media streaming," in *Proc. IEEE INFOCOM 2003*, 2003.
[9] Y. H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE J. Select. Areas Commun.*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.
[10] N. Magharei, D. Stutzbach, and R. Rejaie, "Peer-to-peer receiver-driven mesh-based streaming," in *Proc. ACM SIGCOMM 2005, Poster Session*, Aug. 2005.
[11] C. Dana, D. Li, D. Harrison, and C.-N. Chuah, "Bass: Bittorrent assisted streaming system for video-on-demand," in *Proc.IEEE Int. Workshop Multimedia Signal Process. (MMSP)*, Oct. 2005.
[12] B. Cohen, "Incentives build robustness in bittorrent," in *Proc. P2P Econ. Workshop*, Berkeley, CA, 2003.
[13] I. M. Vlavianos, A. , and M. Faloutsos, "Bitos: Enhancing bittorrent for supporting streaming applications," in *Proc. IEEE Infocom 2006 Global Internet Workshop*, Apr. 2006.

[14] S. Annapureddy, C. Gkantsidis, and P. Rodriguez, "Providing video-on-demand using peer-to-peer networks," in *Proc. Internet Protocol TeleVision (IPTV) Workshop Conjunction WWW'06*, 2006.
[15] T. Small, B. Liang, and B. Li, "Scaling laws and tradeoffs in peer-to-peer live multimedia streaming," in *MULTIMEDIA'06: Proc. 14th Annu. ACM Int. Conf. Multimedia*, Santa Barbara, CA, 2006, pp. 539–548.
[16] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and improving a bittorrent network's performance mechanisms," in *Proc. IEEE Infocom 2006*, Barcelona, Spain, Apr. 2006.
[17] *Azureus*, [Online]. Available: http://azureus.sourceforge.net/
[18] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do incentives build robustness in bittorrent?," in *Proc. NSDI'07*, Apr. 2007.
[19] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE Infocom 2005*, Miami, FL, 2005, vol. 4, pp. 2235–2245.
[20] J. Li, Peerstreaming: A practical receiver-driven peer-to-peer media streaming system msr-tr-2004-101 Microsoft Research, Tech. Rep., Sep. 2004.
[21] B. Bollobas, *Random Graphs*. New York: Cambridge Univ. Press, 2001.
[22] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proc. 32nd ACM Symp. Theor. Comput.*, 2000.
[23] *Networkx*, [Online]. Available: http://networkx.lanl.gov/
[24] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An approach to universal topology generation," in *MASCOTS 2001. Proc. 9th Int. Symp. Modeling, Anal. Simulation Comput. Telecommun. Syst.*, Cincinnati, OH, 2001, pp. 346–353.
[25] S. Choi and C. Kim, "Loss recovery time of impatient variant of tcp newreno," *Electron. Lett.*, vol. 37, pp. 1559–1560, 2001.
[26] S. Guha and P. Francis, "Characterization and measurement of TCP traversal through nats and firewalls," in *Proc. Internet Meas. Conf. (IMC)*, Berkeley, CA, Oct. 2005, pp. 199–211.
[27] B. Ford, D. Kegel, and P. Srisuresh, "Peer-to-peer communication across network address translators," in *Proc. 2005 USENIX Tech. Conf.*, 2005.
[28] B. Ford, "Structures streams: A new transport abstraction," in *ACM SIGCOMM 2007*, Kyoto, Japan.
[29] *Ietf*, [Online]. Available: http://www.ietf.org/html.charters/behave-charter.html

Darshan Purandare (S'07) received the M.S. degree in computer science from the University of Central Florida (UCF), Orlando, in 2005. He is pursuing the Ph.D. degree in the School of Electrical Engineering and Computer Science, UCF.

His research interests include P2P networks, distributed systems, wireless networks, network security, and cryptography.

Ratan Guha (M'90) is a Professor of computer science at the University of Central Florida, Orlando. His research interests include distributed systems, computer networks, security protocols, modeling and simulation, and computer graphics. His research has been supported by grants from the U.S. Army Research Office, U.S. National Science Foundation, STRICOM, PM-TRADE, and the State of Florida.

Dr. Guha is a member of the ACM, IEEE Computer Society, and SCS and has served as a member of the Board of Directors of SCS.