ISSN:0975-9646

Multiplexing in the Peer-to-Peer Live Streaming

T.Ashalatha[1], Ch.Naveen Kumar Reddy[2], G.Vishnu Murthy[3],Ch.Venkata Rajam[4]

[1]Dept. of IT, [2,3,4,] School of Engineering,
Anurag Group of Institutions, Hyderabad, AP, India

Abstract— **The behaviour of commercial peer-to-peer systems for live streaming has been extensively studied in several measurement papers. These studies have to rely on a "black-box" approach, where packet traces are collected from a single or a limited number of measurement points, to infer various properties of traffic on the control and data planes. It is difficult to intuitively understand the observed properties without fully reverse-engineering the underlying systems. In this paper we describe the network architecture of Zattoo, one of the largest production live streaming providers in Europe at the time of writing, and present a large-scale measurement study of Zattoo using data collected by the provider. To highlight, we found that even when the Zattoo system was heavily loaded with as high as 20,000 concurrent users on a single overlay, the median channel join delay remained less than 2 to 5 seconds, and that, for a majority of users, the streamed signal lags over-the-air broadcast signal by no more than 3 seconds**.

Keywords— **Live Streaming, peer-to-peer Systems, network architecture.**

I. Introduction

Live television, time-shifted programming, and content-on-demand are all presently available over the Internet. Increased broadband speed, growth of broadband subscription base, and improved video compression technologies have contributed to the emergence of IPTV services. We draw a distinction between three uses of peer-to-peer(P2P) networks: delay tolerant file download of archival material, delay sensitive progressive download (or streaming) of archival material, and real-time live streaming. In the first case, the completion of download is elastic, depending on available bandwidth in the P2P network. The application buffer receives data as it trickles in and informs the user upon the completion of download. The user can then start playing back the file for viewing in the case of a video file. Bittorrent and variants are example of delay-tolerant file download systems. In the second case, video playback starts as soon as the application assesses it has sufficient data buffered that, given the estimated download rate and the playback rate, it will not deplete the buffer before the end of file. If this assessment is wrong, the application would have to both pause playback and re-buffer, or slow down playback. Video-on-demand systems are examples for this case. The third case, real-time live streaming, has the most stringent delay requirement. While progressive download may tolerate initial buffering of tens of seconds or even minutes, live streaming generally cannot tolerate more than a few seconds of buffering. Taking into account the delay introduced by signal ingest and encoding, and network transmission and propagation, the live streaming system can introduce only a few seconds of buffering time end-to-end and still be considered "live" [1].

The Zattoo peer-to-peer live streaming system was a free-to-use network with a maximum of over 60,000 concurrent users on a single channel.

II. System Architecture

The Zattoo system rebroadcasts live TV, captured from satellites, onto the Internet. The system carries each TV channel on a separate peer-to-peer delivery network and is not limited in the number of TV channels it can carry. Although a peer can freely switch from one TV channel to another, and thereby departing and joining different peer-to-peer networks, it can only join one peer-to-peer network at any one time. We henceforth limit our description of the Zattoo delivery network as it pertains to carrying one TV channel. Fig.1 shows a typical setup of a single TV channel carried on the Zattoo network. TV signal captured from satellite is encoded
into H.264/AAC streams, encrypted, and sent onto the Zattoo network. The encoding server may be physically separated from the server delivering the encoded content onto the Zattoo network. For ease of exposition, we will consider the two as logically co-located on an *Encoding Server*. Users are required to register themselves at the Zattoo website to download a free copy of the Zattoo player application. To receive the signal of a channel, the user first authenticates itself to the Zattoo *Authentication Server*. Upon authentication, the user is granted a ticket with limited lifetime. The user then presents this ticket, along with the identity of the TV channel of interest, to the Zattoo *Rendezvous Server*. If the ticket specifies that the user is authorized to receive signal of the said TV channel, the Rendezvous Server returns to the user a list of peers currently joined to the P2P network carrying the channel, together with a signed channel ticket. If the user is the first peer to join a channel, the list of peers it receives contain only the Encoding Server. The user joins the channel by contacting the peers returned by the Rendezvous Server, presenting its channel ticket, and obtaining the live stream of the channel from them

A. Peer-Division Multiplexing

To minimize per-packet processing time of a stream, the Zattoo protocol sets up a virtual circuit with multiple fan outs at each peer. When a peer joins a TV channel, it establishes a peer-division multiplexing (PDM) scheme amongst a set of neighbouring peers, by building a virtual circuit to each of the neighbouring peers.
Baring departure or performance degradation of a neighbour peer, the virtual circuits are maintained until the joining peer switches to another TV channel. With the virtual circuits set up, each packet is forwarded without further per-packet handshaking between peers.

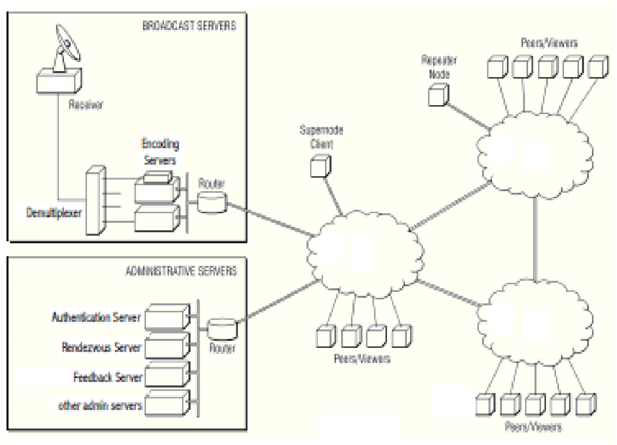

Fig. 1. Zattoo delivery network architecture.

The PDM establishment process consists of two phases: the *search* phase and the *join* phase. In the search phase, the new, joining peer determines its set of potential neighbours. In the join phase, the joining peer requests peering relationships with a subset of its potential neighbours. Upon acceptance of a peering relationship request, the peers become neighbours and
a virtual circuit is formed between them.

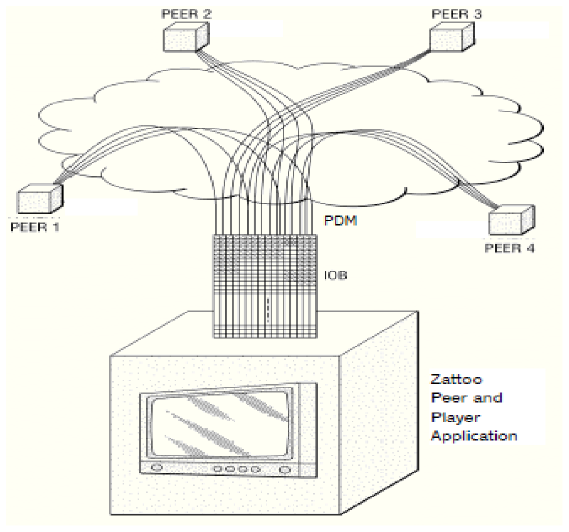

Fig. 2. Zattoo peer with IOB.

B. Stream Management

We represent a peer as a packet buffer, called the IOB, fed by sub-streams incoming from the PDM. The IOB drains to (1) a local media player if one is running, (2) a local file if recording is supported, and (3) potentially other peers. Fig. 2 depicts a Zattoo player application with virtual circuits established to four peers. As packets from each sub-stream arrive at the peer, they are stored in the IOB for reassembly to reconstruct the full stream. Portions of the stream that have been reconstructed are then played back to the user. In addition to providing a reassembly area, the IOB also allows a peer to absorb some variabilities in available network bandwidth and network delay

The IOB is referenced by an *input pointer*, a *repair pointer*, and one or more *output pointers*. The input pointer points to the slot in the IOB where the next incoming packet with sequence number higher than the highest sequence number received so far will be stored. The repair pointer always points one slot beyond the last packet received in order and is used to regulate packet retransmission and adaptive PDM as described later. We assign an output pointer to each forwarding destination. The output pointer of a destination indicates the destination's current forwarding horizon on the IOB. In accordance to the three types of possible forwarding destinations listed above, we have three types of output pointers: *player pointer*, *file pointer*, and *peer pointer*. One would typically have at most one player pointer and one file pointer but potentially multiple concurrent peer pointers, referencing an IOB. The Zattoo player application does not currently support recording

Since we maintain the IOB as a circular buffer, if the incoming packet rate is higher than the forwarding rate of a particular destination, the input pointer will overrun the output pointer of that destination. We could move the output pointer to match the input pointer so that we consistently forward the oldest packet in the IOB to the destination. Doing so, however, requires checking the input pointer against all output pointers on every packet arrival. Instead, we have implemented the IOB as a double buffer. With the double buffer, the positions of the output pointers are checked against that of the input pointer only when the input pointer moves from one sub-buffer to the other. When the input pointer moves from sub-buffer a to sub buffer b, all the output pointers still pointing to sub-buffer b are moved to the start of sub-buffer a and sub-buffer b is flushed, ready to accept new packets. When a sub-buffer is flushed while there are still output pointers referencing it, packets that have not been forwarded to the destinations associate with those pointers are lost to them, resulting in quality degradation. To minimize packet lost due to sub-buffer flushing, we would like to use large sub-buffers. However, the real-time delay requirement of live streaming limits the usefulness of late arriving packets and effectively puts a cap on the maximum size of the sub-buffers.

Different peers may request for different numbers of, possibly non-consecutive, sub-streams. To accommodate the different forwarding rates and regimes required by the destinations, we associate a packet map and forwarding discipline with each output pointer. Fig. 3 shows the packet map associated with an output peer pointer where the peer has requested sub-streams 1, 4, 9, and 14. Every time a peer pointer is repositioned to the beginning of a sub-buffer of the IOB, all the packet slots of the requested sub-streams are marked NEEDed and all the slots of the sub-streams not requested by the peer are marked SKIP. When a NEEDed packet arrives and is stored in the IOB, its state in the packet map is changed to READY. As the peer pointer moves along its associated packet map, READY packets are forwarded to the peer and their states changed to SENT. A slot marked NEEDed but not READY, such as slot n + 4 in Fig.3, indicates that the packet is lost or will arrive out-of-order and is bypassed. When an out-of-order packet arrives, its slot is changed to READY and the peer pointer is reset to point to this slot. Once the out-of-order packet has been sent to the peer, the peer pointer will move forward, bypassing all SKIP, NEED, and SENT slots until it reaches the next READY slot, where it can resume sending.

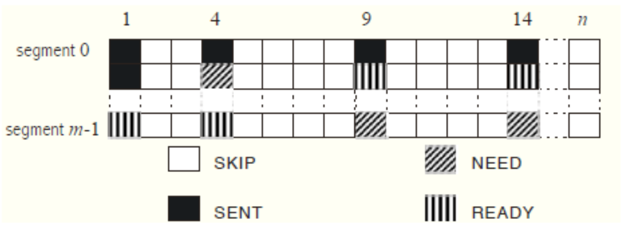

Fig. 3. Packet map associated with a peer pointer.

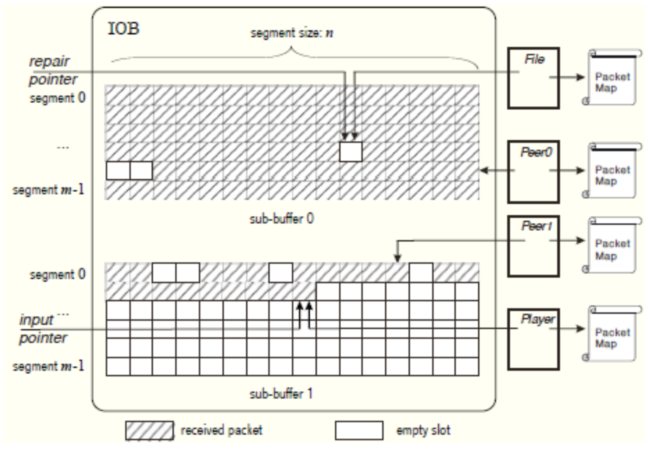

Fig. 4. IOB, input/output pointers and packet maps.

Fig.4 shows an IOB consisting of a double buffer, with an input pointer, a repair pointer, and an output file pointer, an output player pointer, and two output peer pointers referencing the IOB. Each output pointer has a packet map associated with it. For the scenario depicted in the figure, the player pointer tracks the input pointer and has skipped over some lost packets. Both peer pointers are lagging the input pointer, indicating that the forwarding rates to the peers are bandwidth limited. The file pointer is pointing at the first lost packet. Archiving a live stream to file does not impose real-time delay bound on packet arrivals. To achieve the best quality recording possible, a recording peer always waits for retransmission of lost packets that cannot be recovered by error correction.

C. Adaptive PDM

While we rely on packet retransmission to recover from transient congestions, we have two channel capacity adjustment mechanisms to handle longer-term bandwidth fluctuations. The first mechanism allows a *forwarding* peer to adapt the number of sub-streams it will forward given its current available bandwidth, while the second allows the *receiving* peer to switch provider at the sub-stream level. Peers on the Zattoo network can redistribute a highly variable number of sub-streams, reflecting the high variability in uplink bandwidth of different access network technologies. For a full-stream consisting of sixteen *constant*-bit rate sub streams, our prior study show that based on realistic peer characteristics measured from the Zattoo network, half of the peers can support less than half of a stream, 82% of peers can support less than a full-stream, and the remainder can support up to ten full streams (peers that can redistribute more than a full stream is conventionally known as *super nodes* in the literature) [5].

With *variable*-bit rate streams, the bandwidth carried by each sub-stream is also variable. To increase peer bandwidth usage, without undue degradation of service, we instituted measurement-based admission control at each peer. In addition to controlling resource commitment, another goal of the measurement-based admission control module is to continually estimate the amount of available uplink bandwidth
at a peer.

III. Global Bandwidth Subsidy System

Each peer on the Zattoo network is assumed to serve a user through a media player, which means that each peer must receive, and can potentially forward, all n sub-streams of the TV channel the user is watching. The limited redistribution capacity of peers on the Zattoo network means that a typical client can contribute only a fraction of the sub streams that make up a channel. This shortage of bandwidth leads to a global bandwidth deficit in the peer-to-peer network Whereas bittorrent like delay-tolerant file downloads or the delay-sensitive progressive download of video-on-demand applications can mitigate such global bandwidth shortage by increasing download time, a live streaming system such as Zattoo's must subsidize the bandwidth shortfall to provide real-time delivery guarantee.

Zattoo's *Global Bandwidth Subsidy System* (or simply, the *Subsidy System*), consists of a global bandwidth monitoring subsystem, a global bandwidth forecasting and provisioning subsystem, and a pool of *Repeater nodes*. The monitoring subsystem continuously monitors the global bandwidth requirement of a channel. The forecasting and provisioning sub system projects global bandwidth requirement based on measured history and allocates Repeater nodes to the channel as needed.

The monitoring and provisioning of global bandwidth is complicated by two highly varying parameters over time, client population size and peak streaming rate, and one varying parameter over space, available uplink bandwidth, which is network-service provider dependent. Forecasting of bandwidth requirement is a vast subject in itself.

When a bandwidth shortage is projected for a channel, the Subsidy System assigns one or more Repeater nodes to the channel. Repeater nodes function as bandwidth multiplier, to amplify the amount of available bandwidth in the network. Each Repeater node serves at most one channel at a time; it joins and leaves a given channel at the behest of the Subsidy System. Repeater nodes receive and serve all n sub-streams of the channel they join, run the same PDM protocol, and are treated by actual peers like any other peers on the network; however, as bandwidth amplifiers, they are usually provisioned to contribute *more* uplink bandwidth than the download bandwidth they consume. The use of Repeater nodes makes the Zattoo network a hybrid P2P and content distribution network. We next describe the bandwidth monitoring subsystem of the Subsidy System, followed by design of the simple bandwidth projection and Repeater node assignment subsystem.

A. Global Bandwidth Measurement

The *capacity metric* of a channel is the tuple (D,C), where D is the aggregate download rates required by all users on the channel, and C is the aggregate upload capacity of those users. Usually C < D and the difference between the two is the global bandwidth deficit of the channel. Since channel viewership changes over time as users join or leave the channel, we need a scalable means to measure and update the capacity metric. We rely on aggregating the capacity metric up the peer division multiplexing tree. Each peer in the overlay periodically aggregates the capacity metric reported by all its downstream receiver peers, adds its own capacity measure (D,C) to the aggregate, and forwards the resulting capacity metric upstream to its forwarding peers. By the time the capacity metric percolates up to the Encoding Server, it contains the total download and upload rate aggregates of the whole streaming overlay. The Encoding Server then simply forwards the obtained (D,C) to the Subsidy Server

B. Global Bandwidth Projection and Provisioning

For each channel, the Subsidy Server keeps a history of the capacity metric (D,C) reports received from the channel's Encoding Server. The *channel utilization ratio* (U) is the ratio D over C. Based on recent movements of the ratio U, we classify the *capacity trend* of each channel into the following four categories.

Stable: the ratio U has remained within [S-_, S+_] for the past Ts reporting periods.

Exploding: the ratio U increased by at least E between Te (e.g., Te = 2) reporting periods.

Increasing: the ratio U has steadily increased by I (I _ E) over the past T_i reporting periods.

Falling: the ratio U has decreased by F over the past T_f reporting periods.

Orthogonal to the capacity-trend based classification above, each channel is further categorized in terms of its capacity utilization ratio as follows.

Under-utilized: the ratio U is below the low threshold L, e.g., U_0.5.

Near Capacity: the ratio U is almost 1.0, e.g., U_0.9. If the capacity trend of a channel has been "Exploding," one or more Repeater nodes will be assigned to it immediately. If the channel's capacity trend has been "Increasing," it will be assigned a Repeater node with a smaller capacity. The goal of the Subsidy System is to keep a channel below "Near Capacity." If a channel's capacity trend is "Stable" and the channel is "Under-utilized," the Subsidy System attempts to free Repeater nodes (if any) assigned to the channel. If a channel's capacity utilization is "Falling," the Subsidy System waits for the utilization to stabilize before reassigning any Repeater nodes.

IV. Server-Side Measurements

In the Zattoo system, two separate centralized collector servers collect usage statistics and error reports, which we call the "stats" server and the "user-feedback" server respectively. The "stats" server periodically collects aggregated player statistics from individual peers, from which full session logs are constructed and entered into a session database. The session database gives a complete picture of all past and present sessions served by the Zattoo system. A given database entry contains statistics about a particular session, which includes join time, leave time, uplink bytes, download bytes, and channel name associated with the session. We study the sessions generated on three major TV channels from three different countries (Germany, Spain, and Switzerland), from June 1st to June 30th, 2008. We label those channels from Germany, Spain, and Switzerland as *ARD*, *Cuatro*, and *SF2*, respectively. Euro 2008 games were held during this period, and those three channels broadcast a majority of the Euro 2008 games including the final match. See Table I for information about the collected session data sets.

TABLE I
SESSION DATABASE (6/1/2008–6/30/2008).

Channel	# sessions	# distinct users
ARD	2,102,638	298,601
Cuatro	1,845,843	268,522
SF2	1,425,285	157,639

TABLE II
FEEDBACK LOGS (6/20/2008–6/29/2008).

Channel	# feedback logs	# sessions
ARD	871	1,253
Cuatro	2,922	4,568
SF2	656	1,140

The "user-feedback" server, on the other hand, collects users' error logs submitted asynchronously by users. The "user feedback" data here is different from peer's quality feedback used in PDM reconfiguration. Table II describes the feedback logs collected from June 20th to June 29th. A given feedback log can contain multiple sessions (for the same or different channels), depending on user's viewing behaviour. The second column in the table represents the number of feedback logs which contain at least one session generated on the channel listed in the corresponding entry in the first column. The numbers in the third column indicate how many distinct sessions generated on said channel are present in the feedback logs.

TABLE III
AVERAGE SHARING RATIO.

Channel	Average sharing ratio	
	Off-peak	Peak
ARD	0.335	0.313
Cuatro	0.242	0.224
SF2	0.277	0.222

Table III shows the average sharing ratio in the two time periods separately. Zattoo's usage during peak hours typically accounts for about 50% to 70% of the total usage of the day. According to the table, the average sharing ratio during peak hours is slightly lower than, but not very much different from during off-peak hours. *Cuatro* channel in Spain exhibits relatively lower sharing ratio than the two other channels. One bandwidth test site [6] reports that average uplink bandwidth in Spain is about 205 kbps, which is much lower than in Germany (582 kbps) and Switzerland (787 kbps). The lower sharing ratio on the Spanish channel may reflect regional difference in residential access network provisioning. The balance of the required bandwidth is provided by Zattoo's Encoding Server and Repeater nodes.

V. CLIENT-SIDE MEASUREMENTS

To further study the P2P overlay beyond details obtainable from aggregated session-level statistics, we run several modified Zattoo clients which periodically retrieve the internal states of other participating peers in the network by exchanging SEARCH/JOIN messages with them. After a given probe session is over, the monitoring client archives a log file where we can analyze control/data traffic exchanged and detailed protocol behaviour. We run the experiment during Zattoo's live coverage of Euro 2008 (June 7th to 29th). The monitoring clients tuned to game channels from one of Zattoo's data centres located in Switzerland while the games were broadcast live. The data sets presented in this paper were collected during the coverage of the championship final on two separate channels: *ARD* in Germany and *Cuatro* in Spain. Soccer teams from Germany and Spain participated in the championship final.

A. Sub-Stream Synchrony

To ensure good viewing quality, peer should not only obtain all necessary sub-streams (discounting redundant sub streams), but also have those sub-streams delivered temporally synchronized with each other for proper online decoding. Receiving out-of-sync sub-streams typically results in pixelated screen on the player.

B. Peer Synchrony

While sub-stream synchrony tells us stream quality different peers may experience, "peer synchrony" tells us how varied in time peers' viewing points are. With small scale P2P networks, all participating peers are likely to watch live streaming roughly synchronized in time. However, as the size of the P2P overlay grows, the viewing point of edge nodes may be delayed significantly compared to those closer to the Encoding Server. In the experiment, we define the *viewing point* of a peer as the median of the latest sequence numbers across its sub-streams. Then we choose one peer (e.g., a Repeater node directly connected to the Encoding Server) as a reference point, and compare other peers' viewing point against the reference viewing point.

C. Effectiveness of ECC in Isolating Loss

Now we investigate the effects of overlay sizes on the performance scalability of the Zattoo system. Here we focus on client-side quality (e.g., loss rate). As described in Section 1, Zattoo-broadcast media streams are RS encoded, which allows peers to reconstruct a full stream once they obtain at least k of n sub-streams. Since the ECC-encoded stream reconstruction occurs hop by hop in the overlay, it can mask sporadic packet losses, and thus prevent packet losses from being propagated throughout the overlay at large.

VI. PEER-TO-PEER SHARING RATIO

The premise of a given P2P system's success in providing scalable stream distribution is sufficient bandwidth sharing from participating users [5]. Section IV-A shows that the average sharing ratio of Zattoo users ranges from 0.2 to 0.35, which translates into bandwidth uplinks ranging from 100 kbps to 175 kbps. This is far lower than the numbers reported as typical uplink bandwidth in countries where Zattoo is available [6]. Aside from the possibility that user's bandwidth may be shared with other applications, we find factors such as users' behaviour, support for variable-bit rate encoding, and heterogeneous NAT environments contributing to suboptimal sharing performance. Designers of P2P streaming systems must pay attention to these factors to achieve good bandwidth sharing. However, one must also keep in mind that improving bandwidth sharing should not be at the expense of compromised user viewing experience, e.g., due to more frequent uplink bandwidth saturation.

User churns: It is known that frequent user churns accompanied by short sessions, as well as flash crowd behaviour pose a unique challenge in live streaming systems [7], [8]. User churns can occur in *both* upstream (e.g., forwarding peers) and downstream (e.g., receiver peers) connectivity on the overlay. While frequent churns of forwarding peers can cause quality problems, frequent changes of receiver peers can lead to *under*-utilization of a forwarding peer's uplink capacity.

Variable-bit rate streams: Variable-bit rate (VBR) encoding is commonly used in production streaming systems, including Zattoo, due to its better quality-to-bandwidth ratio compared to constant-bit rate encoding. However, VBR streams may put additional strain on peer's bandwidth contribution and quality optimization. As described in Section II-C, the presence of VBR streams require peers to perform measurement-based admission control when allocating resources to set up a virtual circuit. To avoid degradation of service due to overloading of the uplink bandwidth, the measurement-based admission control module must be conservative both in its reaction to increases in available bandwidth and its allocation of available bandwidth to newly joining peers. This conservativeness necessarily lead to underutilization of resources.

NAT reachability: Asymmetric reachability imposed by prevalent NAT boxes adds to the difficulty in achieving full utilization of user's uplink capacity. Zattoo delivery system supports 6 different NAT configurations: open host, full cone, IP-restricted, port-restricted, symmetric, and UDP-disabled, listed in increasing degree of restrictiveness. Not every pair wise communication can occur among different NAT types. For example, peers behind a port-restricted NAT box cannot communicate with those of symmetric NAT type (we documented a NAT reachability matrix in an earlier work [5]).

To examine how the varied NAT reachability comes into play as far as sharing performance is concerned, we performed the following two controlled experiments. In both experiments, we run four Zattoo clients, each with a distinct NAT type, tuned to the same channel concurrently for 30 minutes. In the first experiment, we fixed the maximum allowable uplink capacity (denoted "*max cp*") of those clients to the same constant value. In the second experiment, we let the *max cp* of those clients self-adjust over time (which is the default setting of Zattoo player). In both experiments, we monitored how the uplink bandwidth utilization ratio (i.e., ratio between actively used uplink bandwidth and maximum allowable bandwidth *max cp*) changes over time. The experiments were repeated three times for reliability.

VIII. CONCLUSION

We have presented a receiver-based, peer-division multiplexing engine to deliver live streaming content on a peer-to-peer network. The same engine can be used to transparently build a hybrid P2P/CDN delivery network by adding Repeater nodes to the network. By analyzing large amount of usage data collected on the network during one of the largest viewing event in Europe, we have shown that the resulting network can scale to a large number of users and can take good advantage of available uplink bandwidth at peers. We have also shown that error-correcting code and packet retransmission can help improve network stability by isolating packet losses and preventing transient congestion from resulting in PDM reconfigurations. We have further shown that the PDM and adaptive PDM schemes presented have small enough overhead to make our system competitive to digital satellite TV in terms of channel switch time, stream synchronization, and signal lag

ACKNOWLEDGMENT

This work is supported by the Anurag Group of Institutions, AP, India, Prof.M.Mutha Reddy, Dean, School of Engineering, Hyderabad, India

REFERENCES

[1] R. auf der Maur, "Die Weiterverbreitung von TV- und Radioprogrammen ¨uber IP-basierte Netze," in *Entertainment Law (f. d. Schweiz)*, 1st ed. St¨ampfli Verlag, 2006.
[2] UEFA, "Euro2008," http://www1.uefa.com/.
[3] S. Lin and D. J. Costello, Jr., *Error Control Coding*, 2nd ed. Pearson Prentice-Hall, 2004.
[4] S. Xie, B. Li, G. Y. Keung, and X. Zhang, "CoolStreaming: Design, Theory, and Practice," *IEEE Trans. on Multimedia*, vol. 9, no. 8, December 2007.
[5] K. Shami *et al.*, "Impacts of Peer Characteristics on P2PTV Networks Scalability," in *Proc. IEEE INFOCOM Mini Conference*, April 2009.
[6] Bandwidth-test.net, "Bandwidth test statistics across different countries,"http://www.bandwidth-test.net/stats/country/.
[7] 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, International World Wide Web Conference*, 2006.
[8] B. Li *et al.*, "An Empirical Study of Flash Crowd Dynamics in a P2P-Based Live Video Streaming System," in *Proc. IEEE GlobalTelecommunications Conference*, 2008.
[9] J. R. et al, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)," 1993, RFC 3489.
[10] A. Ali, A. Mathur, and H. Zhang, "Measurement of Commercial Peerto- Peer Live Video Streaming," in *Proc. Workshop on Recent Advances in Peer-to-Peer Streaming*, August 2006.
[11] L. Vu *et al.*, "Measurement and Modeling of a Large-scale Overlay for Multimedia Streaming," in *Proc. Int'l Conf. on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, 2007.
[12] T. Silverston and O. Fourmaux, "Measuring P2P IPTV Systems," in *Proc. ACM NOSSDAV*, November 2008.
[13] C. Wu, B. Li, and S. Zhao, "Magellan: Charting Large-Scale Peer-to-Peer Live Streaming Topologies," in *Proc. ICDCS'07*, 2007, p. 62.
[14] M. Cha *et al.*, "On Next-Generation Telco-Managed P2P TV Architectures,"in *Proc. Int'l Workshop on Peer-to-Peer Systems*, 2008.
[15] D. Ciullo *et al.*, "Understanding P2P-TV Systems Through Real Measurements,"in *Proc. IEEE GLOBECOM*, November 2008.
[16] E. Alessandria *et al.*, "P2P-TV Systems under Adverse Network Conditions:a Measurement Study," in *Proc. IEEE INFOCOM*, April 2009.
[17] S. Banerjee *et al.*, "Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications," in *Proc. IEEE INFOCOM*,2003.
[18] D. Tran, K. Hua, and T. Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming," in *Proc. IEEE INFOCOM*, 2003.
[19] D. Kostic *et al.*, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," in *Proc. ACM SOSP*, Bolton Landing, NY, USA, October 2003.
[20] R. Rejaie and S. Stafford, "A Framework for Architecting Peer-to-Peer Receiver-driven Overlays," in *Proc. ACM NOSSDAV*, 2004..