KQStream: Kindred-Based QoS-Aware Live Media Streaming in Heterogeneous Peer-to-Peer Environments

Yuan He[1], Tao Gu[2], Jiang Guo[3], Jingyao Dai[1]
[1] *Hong Kong University of Science and Technology, Hong Kong*
{heyuan, daijy}@cse.ust.hk
[2] *Institute for Infocomm Research, Singapore*
tgu@i2r.a-star.edu.sg
[3]*Department of Electrical and Computer Engineering, University of Toronto, Ontario, Canada*
jguo@eecg.toronto.edu

Abstract

*This paper presents the design of a **K**indred-based **Q**oS-aware Live Media **Stream**ing (**KQStream**) system in heterogeneous peer-to-peer environments. Motivated by the Multiple Description Coding technique, we propose a novel kindred-based approach to construct a peer-to-peer overlay topology. To achieve QoS-awareness during the streaming process, we propose a dynamic QoS-aware regulating technique which takes into account of user's preference, network bandwidth and buffer size. With the integration of the two proposed techniques, KQStream provides more flexibility in heterogeneous peer-to-peer environments and configurable playback with dynamic QoS-aware regulating.*

1. Introduction

With widespread broadband penetration, multimedia services become increasingly popular among Internet users in recent years. However, the rapid development of broadband applications and multimedia proliferates beyond the capacity of the traditional server-client architecture. The problems are two folds: (i) the simple pileup of hardware on servers will never suffice to meet the rapidly increasing user demands; (ii) the dynamic boosting of network connections which occurs commonly during a popular live streaming program, such as a soccer match, causes network bottlenecks at the server, or in the region with a high density of users. The former limits the scalability of a media streaming system; and the later results in poor stability.

Peer-to-Peer (P2P) media streaming services are proposed to overcome some of these limitations and become popular in recent years. The feasibility of supporting large-scale groups using the application end-point architecture was proven in [1]. Following the *overlay multicast* technology, many models and systems have been proposed [3, 4, 5, 6, 7]. In [6], cooperative streaming is advocated to ease the burden on central servers. The tree structure is constructed on the basis of the global knowledge from a central server. In [4], CoolStreaming is proposed to explore collaborations among peers by dividing the streaming buffer into equivalent segments, keeping the bitmap mapping information and exchanging it with a gossip-like method. Based on the principle in CoolStreaming, a number of peer-to-peer live media streaming systems have been proposed [19, 20] and gained popularity.

In this paper, we propose a **K**indred-based **Q**oS-aware live media **Stream**ing (**KQStream**) system. We aim to improve a streaming system from two aspects: (i) Flexibility in heterogeneous peer-to-peer environments; and (ii) Configurable playback process with dynamic QoS-aware regulating, which is close to a user's real experience. We adopt the multi-tree structure [1] along the streaming route to build the overlay topology for KQStream. Peers exchange the kindred information with their neighbors, including their parents, children, siblings and their descriptions.

Motivated by the recently developed video coding techniques such as [13, 14], we propose a *Multiple Description Coding (MDC)* [14] based method for P2P live streaming. The MDC based method improves not only the adaptability of a media streaming system but also the flexibility of peers to configure the streaming process of their own.

QoS mechanisms may be difficult to implement in a heterogeneous P2P environment due to the complexity of measuring and control of diverse connections [11, 16, 17]. To achieve QoS-awareness during the streaming process, we propose a dynamic QoS-aware regulating technique which takes into account of user's preference, network bandwidth and buffer size. Based on the QoS-aware mechanism, the playback process

can then be designed to match a user's experience such as the requirements of media quality, playback stability and continuity.

The rest of this paper is organized as follows: We present the kindred-based topology construction in Section 2 and the QoS-aware mechanism on heterogeneous peers in Section 3. Section 4 presents the evaluation results. We conclude the paper in Section 5.

2. Kindred-based Topology Construction

We encode the original media into multiple descriptions. Therefore, they can be distributed in multiple streaming routes. Since the number of descriptions each peer demands may vary due to diverse network conditions, KQStream complements the traditional multi-tree topology with a kindred-based node structure.

2.1. Node Structure

To illustrate the node structure, we take an example topology consisting of 5 nodes. As shown in Fig. 1, arrows represent the streaming routes for different descriptions. Tree a, tree b and tree c respectively deliver descriptions "a", "b" and "c". Siblings share same descriptions from a common parent, such as nodes C and E in tree a.

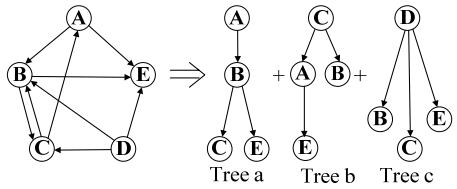

Figure 1: Multi-tree topology in KQStream

```
struct Kindred_info
{
   UINT NodeID; //Representative ID of a node;
   UINT des; //Description shared with current kindred;
   int depth;
   //Overlay Hops the description has been passed
  // from the original source.
  int relation;
  //0:candidate; 1:parent; 2:child; 3:sibling;
};
```

Figure 2: Kindred_info structure in KQStream

We define the structure of a node as follows: Each node holds a set (named *KindredSet*) of *Kindred_info* entries, which store the kindred information between the current node and all its parents, children and siblings, as shown partially in Fig. 2. While a child is obtaining descriptions from its parent, it dispatches them at the same time. A peer, together with its parent, siblings and children, logically form a *KindredSet* which shares a common description. By exchanging the information about the *KindredSet*, the information of available descriptions is diffused within the scope of neighborhood. After a node exchanges the *KindredSets* with its neighbors, not only the description information on the neighbor nodes is obtained, but also the available resources on the neighbor node's kindred can be indirectly located. Note that there might be an anonymity issue [10] in such information exchange. The further discussion is out of scope of this paper.

2.2. Node Dynamics

2.2.1 Node joining. We illustrate a typical streaming request process during node joining in Fig. 3. The arrows in different types of lines represent the connections of different descriptions. Upon joining, node P first receives a list of IP addresses of its neighbors (nodes A, B, C and D) and their associated media information through a bootstrap server. It shapes its streaming requirement with the considerations of the user's preferences, available bandwidth, and buffer size. It then notifies each of its neighbors with a JOIN message; and each neighbor responses with a packet containing their *KindredSets*.

Subsequently, node P builds its own *KindredSet* in which each entry corresponds to an available description and its resident node called a candidate. The streaming request process is able to start right after a certain number of packets are received, i.e. node P need not wait until all the *KindredSets* of its neighbors are received. Late arrived packets are reserved for future requests or in case of node dynamics.

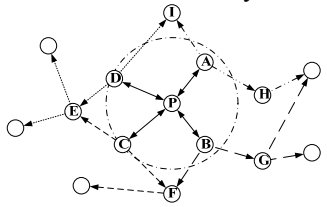

Figure 3: Exchanging Kindred information

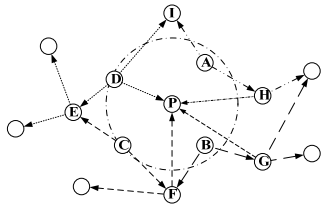

Figure 4: Requesting streaming

2.2.2. Peer selection. When a node selects source peers of the requested descriptions, its parents are sorted out from the candidates according to certain metrics, as we discuss later in this paper. A brief content of the metrics includes: link delay, contribution degree of the candidate, tree depth, etc.

To illustrate the selection process, nodes D, F, G and H are selected as P's parents as shown in Fig. 4. If there is available upload bandwidth, they responds positively to the streaming requests from P. Streaming connections are then established. Node P and its parents register relevant *Kindred_info* entries into their *KindredSets*, and their siblings who share the common parents with node P are added into its *KindredSet*.

2.2.3. Peer leaving. When a peer leaves the network, it first sends a *QUIT* message to all its kindred. Upon receiving the message, all its kindred cease the streaming connection with the leaving peer, update their local *KindredSets* and release occupied resources.

In case of unexpected failure or network outage, a peer does not send any message to its kindred. We use periodical probing to detect a leaving peer under these circumstances. All living peers in the network ping the entries in their *KindredSets* periodically. The entry with a pong response will remain; otherwise will be left out in a *KindredSet*.

2.3. Optimizations

Topology optimization can improve the overlay performance [9, 15]. We propose three optional policies for the kindred-based topology optimization.

2.3.1. Load balancing. Without an appropriate load balancing technique, a few strong nodes with predominant network conditions will probably attract a large number of demands resulting in a heavy-tailed topology. Abstemious utilization of their bandwidth increases the availability of descriptions within its neighboring area.

To balance load during the streaming process, we introduce a parameter named *Contribution-degree*, defined as follows:

$$Contribution\text{-}degree = U_C / U_T \quad (1)$$

U_C: uploading bandwidth in use of the current peer. U_T: total available uploading bandwidth of the peer. Candidates with lower *Contribution-degree* will be sorted out first. In this way we reasonably balance the load on multiple trees.

2.3.2. Shorten tree depth. To minimize the maintenance cost of the overlay, a tree topology with a shorter depth is desired. To this goal, we introduce a parameter in the *Kindred_info* structure to represent the number of overlay hops that the description has been passed from the original source, as shown in Fig. 2. Descriptions of the shortest depth are requested first. Thus, the newly joining peers are inclined to be positioned near the root (i.e., the video source). Thus a tree topology with a shorter depth can be generated.

2.3.3. Shorter delay is favorable. A shorter link delay between peers reflects better connectivity on the whole. In addition, for a client receiving multiple descriptions in live media streaming, convergent link delay from multiple sources could improve the efficiency of MDC-decoding process.

3. QoS-aware Streaming

The QoS-aware mechanism in KQStream is based on the individual streaming state of peers, in other words, is end-hosts/users oriented. We consider a user's experience during the playback process in our design. Firstly, different from the existing models which provide uniform streaming at a simplex rate, KQStream allows users to configure their playback requirements. Secondly, we use a re-buffering technique for the QoS regulating of dynamic connections. Under certain network conditions, the re-buffer frequency can be determined by a user's choice between the media quality and playback continuity in his/her preference. Thirdly, we use a dynamic QoS-aware regulating method during a streaming process according to a user's requirements and network resources. We describe each of them in details in the remaining of this section.

3.1. Pre-streaming Configurations

3.1.1 Playback rate. In KQStream, peers follow a rule called *demand according to ability* for streaming configurations. Based on this rule, a peer should not request for media quality over its available downloading bandwidth.

$rate_{max}$: potential maximum downloading bandwidth.

$rate_{min}$: minimum acceptable playback rate.

$rate_{set}$: actual appointed playback rate.

p: user's preference of media quality. $0 \le p \le 1$.

p denotes a user's choice between the media quality and playback continuity in his/her preference.

$$rate_{set} = p*rate_{max}+(1-p)*rate_{min} \quad (2)$$

We can obtain good media quality with a higher value of p. However, excessive requirement on media quality will probably cause heavy burden on links.

3.1.2. Re-buffering threshold. A user usually has certain requirements for the streaming quality. When the ratio of received descriptions to the total required quantity is less than a threshold, a re-buffering is triggered at the current peer.

$thres_{set}$: the threshold to trigger re-buffering.

$thres_{hi}$ & $thres_{lo}$: predefined upper and lower bounds of $thres_{set}$.

q: user's preference of streaming quality. $0 \le q \le 1$.

$$thres_{set} = q*thres_{hi}+(1-p)* thres_{lo} \quad (3)$$

When the value of q is higher, lower streaming quality is avoided. And if it is set too high, the increased frequency of re-buffering will obviously result in decreased continuity of playback.

3.1.3. Buffer to cache descriptions. We first define the parameter *bsize* as the available buffer size to cache descriptions. The occupied buffer size for caching is then a product of the quantity (n_b) and the average length of cached descriptions (l_b). Suppose constant s is the unit size of 1 second description:

$$bsize = n_b* l_b* s \quad (4)$$

Caching a long period of a description may decrease the bandwidth utilization, but it ensures the continuity and stability of the playback. On the other hand, caching more descriptions helps to avoid the quality fluctuation and may lead to better flexibility in highly dynamic environments. As a peer usually has a limited buffer size, it's necessary to find the tradeoff between the above two strategies and we will address it in our future work.

3.2 QoS-aware Regulating

3.2.1 Regulating towards link dynamics. During a playback, a peer monitors the loss rate on each streaming link. If the loss rate of a link exceeds an acceptable level, the actual rate playback ($rate_{act}$) decreases.

$$ratio = rate_{act} / rate_{set} \quad (5)$$

The system performs the following regulating:

- $ratio = 1$, keep streaming and playback in normal state.
- $thres_{set} < ratio < 1$, keep playback and try to reconnect with the failed link(s).
- $ratio < thres_{set}$, re-buffering is triggered. The peer suspends playback but keeps downloading descriptions from its parents. Simultaneously it tries to reconnect from the failed links. As soon as $rate_{act}$ reaches $rate_{set}$, the playback resumes.

To improve the efficiency of topology construction, we limit the time of the attempts with a same candidate to a short period. The peer chooses substitute source from its possible candidates if it exceeds the period.

TABLE 1: Number of Parents

200 peers in total		Number of Descriptions		
		4	6	8
Number of Parents	1	40	23	24
	2	63	43	31
	3	71	56	37
	4	26	47	48
	5	--	19	30
	6	--	12	17
	7	--	--	10
	8	--	--	3
	Total	**200**	**200**	**200**

3.2.2. Regulating towards variable buffer size. From the discussion of (4), we conclude that either a longer period of a description or more descriptions will improve the streaming quality. However, both of them are buffer-consuming, and they are constrained by limited buffer space. When *bsize* decreases, a peer first gives up reserving those descriptions content which have already been played back. If it is still not enough to mitigate the shortage, either n_b or l_b will be reduced. Which factors should be sacrificed depends on the user's preferences between playback stability and system flexibility, as we discussed right after (4).

4. Evaluation

4.1. Overlay Radius

In KQStream, the paths for propagating the descriptions are modeled as a Breath-First Search (BFS) tree. A node has multiple representatives in the tree due to the streaming of multiple descriptions. Fig. 5 shows a BFS tree, for the overlay topology shown in Fig. 1. We take an approximate analogism from the homogenous environments. Suppose the average number of direct kindred (i.e., parents or children) of a peer is M in KQStream. An internal node in a BFS tree thus has $M-1$ children. We get a logarithmic relation (Similar but more detailed demonstration is presented in [4]) between the depth (d) and the number of nodes (N) in the BFS tree:

$$d < log_{M-1}N + 3 \quad (6)$$

This reveals a logarithmic relation between the overlay radius and its size, which implies the scalability of KQStream. We conducted experiments to count the peer distribution over the numbers of parents. As shown in Table 1, most peers have more than one parent, and very few peers get only one description from each parent, as indicates that KQStream maintains stable and effective "multi-tree" topologies.

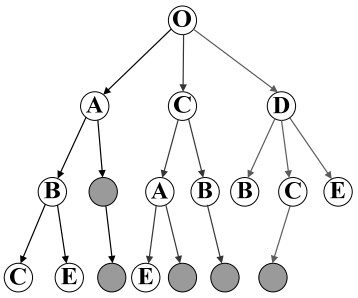

Figure 5: A BFS tree of 5 nodes

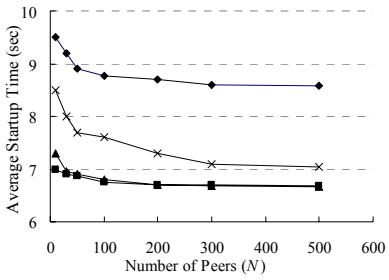

Figure 6: Statistical results of average startup time (*fp* = 3%)

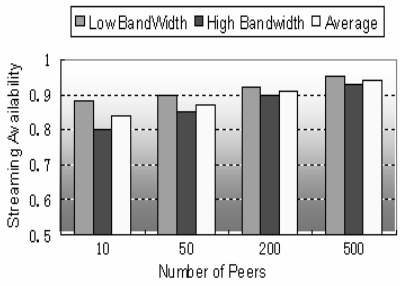

Figure 7: Streaming availability as Num_{des}=6, *fp*=3%

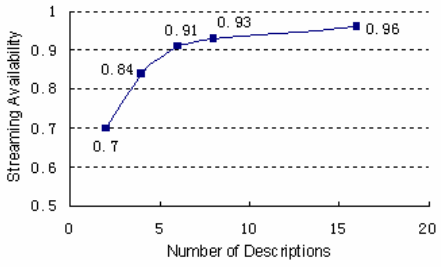

Figure 8: Average streaming availability as *N=200, fp*=3%

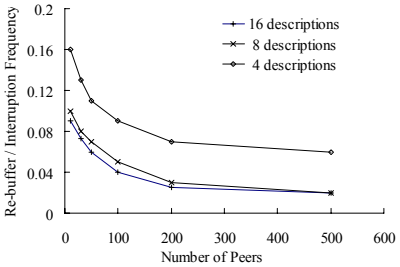

Figure 9: Re-buffer/ Interruption frequency as *fp*=3%

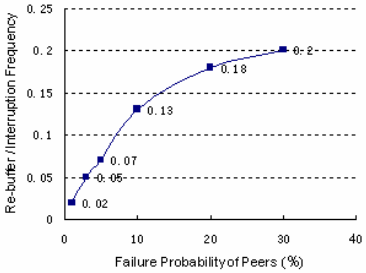

Figure 10: Re-buffer Interruption frequency as *N*=100, Num_{des}=8

4.2 Simulations

We develop a simulator to evaluate KQStream. We use BRITE [18] generate randomized AS (Autonomous System) level topologies. We obtain the results from a series of experiments. We use the following metrics which are adopted from the white paper [2] published by Akamai Technologies, Inc.:

rate: the original full playback rate of KQStream.

Num_{des}: the number of MDC-descriptions (channels) the full media streaming is coded to.

N: the number of peers in KQStream.

fp: the preset failure probability of any peer in KQStream.

4.2.1. Startup time. Assume *fp*=3%, we run the simulator for 10 times over different topologies. Each time, the simulator is configured with different value of *N* (10, 30, 50, 100, 200, 300 and 500) and Num_{des} (2, 6, 10 and 16). The results of average startup time in KQStream are shown in Fig. 6.

From the results, we know a proper granularity of MDC is the most important factor to guarantee the client's performance. Fewer descriptions in MDC lead to a bigger size for each description and result in spending more time to download. However, too many descriptions may complicate the streaming requests and the decoding process from multiple sources. Secondly, certain overlay size is necessary for obtaining short startup time. However, when the number of peers increases over a certain value, for example, N=100 and Num_{des}=6, the average startup time keeps at a nearly constant level which indicates a potential limit.

4.2.2. Streaming availability. Streaming availability denotes the proportion of reception to requirement during the course of streaming. Assume *fp*=3%, we repeat the experiments with two settings as follows. The results are obtained from 10 runs for each experiment.

In the first setting, we set Num_{des} to 6, and *N* set to 10, 50, 200 and 500 respectively. We use a bound value (i.e., 300kbps) to classify all the peers according to their initial downloading bandwidth. Fig. 7 shows the separate data together with the total average result. From the results, we conclude that a larger overlay size steadily leads to higher availability of the system; and low-bandwidth nodes usually have better availability than high-bandwidth nodes because they demand less than those high-bandwidth nodes in general.

In the second setting, we set N to 200, and Num_{des} to 2, 4, 6, 8 and 16 respectively. Fig. 8 plots the streaming availability vs. different number of descriptions. From the results, we conclude that fine-grained media coding helps to increase availability.

4.2.3. Re-buffering/Interruption frequency. The re-buffering/interruption frequency denotes the stability and continuity of streaming. Assume fp=3%, we conduct the experiments with different value of N and Num_{des} to measure the re-buffering/interruption frequency. The results are shown in Fig. 9. Similarly, we conclude that increasing number of peers and proper granularity of MDC help to achieve better stability and continuity.

We repeat the above experiment to study the impact of peer dynamics by setting N and Num_{des} to a recognized proper value and preset fp (the failure probability of peers). The result shown in Fig. 10 reveals the following: The failure probability has obvious impact on the system's stability and streaming continuity. However, under normal conditions, when fp<5%, KQStream shows satisfying stability. Even if the failure probability goes up to 30%, which might occasionally happen in highly unstable environments, the performance still remain at a tolerable level.

5. Conclusions and Future work

In this paper, we propose KQStream: kindred-based QoS-aware live media streaming in heterogeneous peer-to-peer environments. Based on MDC, we propose a kindred-based approach to construct a peer-to-peer overlay topology. In addition, we propose a QoS-aware mechanism with the consideration of a user's preference, bandwidth and buffer size. We analyze of the overlay radius topologies proving KQStream is scalable. The simulation results demonstrate the effectiveness of KQStream in terms of response speed, streaming availability and continuity in heterogeneous peer-to-peer environments.

In our future work, we plan to design strategies for automatic QoS-aware streaming configuration and extend KQStream to the media live streaming in mobile P2P environments.

Acknowledgements

This work is supported in part by NSFC grant No. 60673179. We wish to thank Kuien Liu in the Chinese Academy of Sciences for his valuable feedback and suggestion.

References

[1] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H.Zhang, "The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points," in *Proc. ACM SIGCOMM*, Portland OR, August 2004.

[2] "*Akamai Streaming: When Performance Matters*," http://www.akamai.com/en/resources/pdf/whitepapers/Akamai_Streaming_Performance_Whitepaper.pdf

[3] J. Guo, Y. Zhu, and B. Li, "CodedStream: Live Media Streaming with Overlay Coded Multicast", in *Proc. SPIE/ACM MMCN*, pp.28–39, January 2004

[4] X. Zhang, J. Liu, B. Li, and T.P. Yum, "CoolStreaming/DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming," in *Proc. IEEE INFOCOM*, Miami, FL, USA, March 2005

[5] Y. Chu, S. Rao, and H. Zhang, "A Case for End System Multicast," in *Proc. ACM SIGMETRICS*, pp.1–12, June 2000.

[6] V.N. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. NOSSDAV*, Miami Beach ,FL, USA, May 2002.

[7] M. Hefeeda, A. Habib, B. Botev, D. Xu, B. Bhargava, "PROMISE: Peer-to-Peer Media Streaming Using CollectCast," in *Proc. ACM MM*, Berkeley, California, USA, November 2003.

[8] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM*, Pittsburgh, PA, August 2002.

[9] Yunhao Liu, Li Xiao, and Lionel M Ni, "Building a Scalable Bipartite P2P Overlay Network", *IEEE Transactions on Parallel and Distributed Systems,* 2007

[10] Jinsong Han and Yunhao Liu, "Rumor Riding: Anonymizing Unstructured Peer-to-Peer Systems", in *Proc. IEEE ICNP,* 2006.

[11] Yunhao Liu, Zhenyun Zhuang, Li Xiao, and Lionel M Ni, "A Distributed Approach to Solving Overlay Mismatching Problem", in *Proc. IEEE ICDCS,* 2004.

[12] D.A. Tran, K.A. Hua, and T.T. Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming," in *Proc. IEEE INFOCOM*, San Francisco, CA, March-April 2003.

[13] G.J. Conklin, G.S. Greenbaum, K.O. Lillevold, A.F. Lippman, and Y.A. Reznik, "Video Coding for Streaming Media Delivery on the Internet," *IEEE Transactions On Circuits and Systems for Video Technology*, vol.11, no.3, pp.269-281, March 2001.

[14] V.K. Goyal, "Multiple Description Coding: Compression Meets the Network," *IEEE Signal Processing Magazine*, pp.74–93, September 2001.

[15] Yunhao Liu, Xiaomei Liu, Li Xiao , Lionel M Ni, and Xiaodong Zhang, "Location-Aware Topology Matching in P2P Systems", in *Proc. IEEE INFOCOM,* 2004.

[16] B. Li, D. Xu, and K. Nahrstedt, "An integrated runtime QoS-aware middleware framework for distributed multimedia applications," *ACM Multimedia Systems Journal*, vol.8, pp.420–430, 2002

[17] T. Yamazaki, "QoS Adjustment Scheme for Video Transmission to a Heterogeneous Receivers Group", in *Proc. IMMCN*, pp.996-1000, March 2002.

[18] A. Medina, A. Lakhina, I. Matta and J. Byers, "BRITE: Universal Topology Generation from a User's Perspective," Technical Report BUCS-TR2001-003, 2001. http://www.cs.bu.edu/brite/publications/.

[19] PPLive, http://www.pplive.com

[20] SopCast, http://www.sopcast.com