Peer assisted VoD for set-top box based IP network

Vaishnav Janardhan
Department of Computer Science
Columbia University
vj2135@columbia.edu

Henning Schulzrinne
Department of Computer Science
Columbia University
hgs@cs.columbia.edu

Abstract - **IP-enabled set-top boxes are becoming key devices in home entertainment networks. In addition to providing TV signals, STBs have been providing pay-per view service for a long time. But this service suffers from bandwidth requirements at the source server and has scaling problems. We propose a new design for providing a peer-assisted VoD service where peers co-operate in delivering the content to other peers. This design uses a Bittorrent like protocol for information exchange and peer-to-peer topology management with low startup time, provision for VCR operation and admission control to guarantee QoS for subscribers. It utilizes the large storage of STBs for better viewing experience with reduced jitter and the underlying network architecture to do a location aware content fetching and reduce the expensive cross AS traffic over the Internet.**

1. INTRODUCTION

IP-enabled set-top boxex (STBs) have become key devices for home networks, supporting voice, video and data. Cable operators (MSOs) have been providing a limited menu of video-on-demand (VoD) choices on STBs for a while, and are now transitioning to IP-based delivery and set-top boxes that contain large disk drives. For examlple, the most recent model of the Tivo DVR contains a 250 GB drive, enough to store 300 hours of standard-quality MPEG2 or 32 hours of high-definition video. Streaming VoD from centralized servers is expensive, both in terms of bandwidth and server capacity. For example, YouTube is streaming 40 million videos and 200 TB of data each day, paying 1 million dollars each month for transmission. The cost would be considerably higher for high-definition, movie-length content. In 2002, Netflix was said to distribute about 1,500 TB of DVDs each day [1]. Given these scaling problems, we propose a peer-assisted VoD architecture that uses localized p2p systems to stream video to STBs from other STBs in the same local network. Our approach offers low startup delay, reduced dealy during VCR operations, locality-aware content fetching and distributed admission control. The rest of the paper is structured as follows: Section 2 presents other related work in providing P2P VoD service. Section 3 gives an overview of the system; Section 4 gives the new proposed architecture for providing VoD service. Section 5 gives the system design by describing the specifications and interaction between modules. Section 6 gives the scheduling policies for data exchange and geographical aware node selection for low cost data exchange. Section 7 lists the parameters to evaluate the performance of the VoD service and admission control of new incoming nodes.

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

2. RELATED WORK

There has been a significant amount of work done on providing video streaming using p2p networks in both the industry and academia, see [[1], [6], [9],[10],[7] and [3]]. Most of them have either proposed an application level multicast tree based approach or mesh based push or pull approach for distributing the content. In MediaGrid [6], they have designed the VoD service for STBs, but they do not use the large storage of STBs for pre-fetching and do not support VCR operations. Most of these papers [[9], [6]], do not support VCR operations, load balancing or admission control, guaranteed QoS to the subscriber is very important in pay-per view service. In this paper we propose a system which has a low startup time, provides better user experience with reduced jitter by pre-fetching the contents and very low seek delay during VCR operations by prioritizing the download. This architecture combines and extends the ideas and concepts from a number of other papers.

3. SYSTEM OVERVIEW

For our architecture, we make a number of assumptions that motivate our design. We assume that STBs have ample storage; STBs can peer-to-peer stream either movies that the

[1]New York Times, September 23, 2002

owner of the STB has watched himself or those that some coordination function downloads to the STB in anticipation of demand by other viewers. The details of this anticipatory caching of content are left for further study. STBs are generally assumed to be available continuously, even if the owner is not watching a movie. If that it not the case, the number of available servers is reduced, but should still be proportional to the number of active viewers. STBs could be centrally controlled by the content provider for the storing of any data, for any length of time on any node. We also consider that all the nodes have similar upstream and downstream bandwidth capacities, which will be more than the video encoding/streaming rate. All STBs within a geographical area connects to a common aggregation router forming a subnet and these subnets would in turn connect to the national IP backbone network. All STBs within the subnet will have the same IP address prefix, which helps the STB to identify the location of a peer. All these STBs from different subnets will form an unstructured p2p network. As in BitTorrent [5], movies are split into chunks that can be retrieved individually. The chunk size is of secondary importance, but due to the nature of DVD-style navigation, it may make sense to divide movies into DVD chapters, as users are more likely to navigate by chapter than randomly. In modern residential and campus broadband networks, the downstream and upstream bandwidth is generally sufficient to stream at least one movie, i.e., at least 1 Mb/s for TV-quality viewing. If the upstream bandwidth is insufficient to stream one movie, prefetching will still work, but a single viewer will occupy the upstream bandwidth of several peer STBs, thus limiting the maximum number of concurrent viewers during the busy hour. Since the peer-assisted mechanism is operated either by or in cooperation with the STB provider or DVR vendor, we are less concerned about providing incentives to cooperation. However, some form of quid-pro-quo mechanism could easily be added if desired. We assume that the cost of bandwidth is significantly lower within the provider's network compared to long-distance distribution. A local network could be a single subnet or an autonomous system and would typically be roughly the size of the service area of a cable headend. We were unable to find statistics for the sizes of such service areas, but the number of counties in the United States, around 3,000, probably give a good indication of the order of magnitude.

4. ARCHITECTURE FOR P2P VOD

4.1 Overview

The system consists of four modular pieces that can evolve independently: a directory, a chunk map, a retrieval protocol and a video client. The directory maps names of movies to internal identifiers, using a global DHT. For each movie, a chunk map indicates who holds a copy of a chunk within the local network. The chunk map is stored in an unstructured peer-to-peer network since there will likely be many

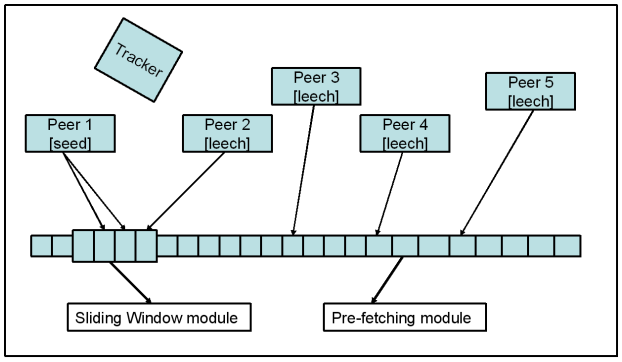

Figure 1: New video-on-demand service architecture

copies of the same chunk. The chunk map is updated as STBs download content. As in [10], we assume that content providers seed a sufficient number of STBs with content, based on estimates of its future popularity. (To deal with flash crowds, content can be pushed to peers ahead of the release date, just like DVDs are shipped to stores today a few days ahead of the official release date. Time-restricted DRM or the later delivery of crucial components, such as parts of MPEG I frames, prevents viewing ahead of the official date.)The retrieval protocol downloads chunks from the peer STBs, as fast as server and network capacity allow. In Section 6.2, we discuss the prioritization of which chunks to download first. Each STB acts as a video server, and responds to standard stream control protocols such as RTSP [8]. The video client itself could well be the same that is used for traditional streaming today, probably drawing on the local STB. (This also allows multiple viewers within the same home.)

The proposed architecture is as shown in the Figure 1. The client running at every node will have a sliding window and pre-fetching module. Both will be simultaneously will be downloading the content based on their priority.

4.2 Sliding-window module

The sliding-window module will be maintaining a sliding window over the video file buffer and will be feeding the chunks needed by the video player. Once the chunks are consumed by the video player, the sliding window moves forward over the video file to fetch new chunks. The chunks that are needed by the sliding window are of high priority and will have to be immediately fetched if not already buffered. The fetching of the chunks would be sequential.

4.3 Pre-fetching module

As the download bandwidth available is more than the video encoding/streaming rate, the pre-fetching module will download the entire file with lower priority, while the sliding window will be downloading the immediately needed chunks with higher priority. The sliding-window and pre-fetching modules are independent of each other and will be buffering the chunks based on their priority. The process of

fetching of chunks has been split up into two different modules for efficient methods of fetching and buffering the entire video file. Both the modules will be buffering the content to the same pool of chunk buffer. When the sliding window module has to feed the video player with the chunks, it first checks if the chunks are available in the local cache. If not available, it will fetch the contents from the set of peer nodes. The fetching of the chunks for the sliding window is done with higher priority. First it will try to retrieve the next sequential chunk from its peers and if it is not available and the chunk is fast approaching the *chunk deadline*, it will go to the source node and fetch the chunk. The sourcing of the chunk from source is done only when it fails to find the chunk among its peers and the *chunk deadline* is fast approaching. The *chunk deadline* is a parameter that is based on a function that compares the expected playback time of the chunk and the minimum time needed to download it. Every chunk in the sliding window will have a different *chunk deadline* and it gets re-computed for all chunks when the sliding window slides ahead.

This model of two modules downloading, ensures the timely delivery of the needed data and the pre-fetching helps in the efficient usage of network resources and brings in more robustness to the p2p network by distributing the content at a faster rate when compared to others models, which do either a sequential fetch of the content or stream it from multiple peers. Due to active pre-fetching by a separate module, the entire video will be buffered in the system during the initial few minutes of the movie, giving the user considerable leeway in doing a random seek without any jitter. Due to active pre-fetching we not only bring in more robustness to the network but also better user experience. Simulations for goodput (index of better user experience, [7]) done in [7] show that during the switching of nodes, from where the data is streamed leads to considerable jitter during switchover and pre-buffering in local nodes will avoid this jitter. Another important requirement of the VoD service is the provision for VCR operations. Consider the scenario when the user is watching the video at the 10^{th} minute of the movie and the sliding window will buffer the immediately needed video chunks. But the user suddenly wants to jump over to the 40^{th} minute of the movie. Now the sliding window module immediately jumps over to the 40^{th} minute of the movie and tries to retrieve the video chunks corresponding to the sliding window's position for the 40^{th} minute of the video. Experiments done on users in [4] show that the number of forward seeks or VCR operation increases with the progress of the session. Active pre-fetching of the entire content would help in serving the increased forward seeks better without any jitter, with the progress of the session.

4.4 Stream Prioritization

As described above, we would like to maximize the amount of data that a viewer can download, as this minimizes the probability of glitches during regular playback and during VCR fast-forward and skip-ahead operations. However, there is the danger of self-interference, i.e., that a lower-priority stream for the same household reduces the throughput of a higher-priority stream if they share the same bottleneck link. We propose three mechanisms to minimize this problem, namely DiffServ, probing and using multiple streams. Using DiffServ, we mark higher lower-priority packets accordingly, so that routers can give packets appropriate scheduling and dropping priority. Many home routers support DiffServ and some 802.11 access points also use DiffServ markings for prioritization, using 802.11e. However, it is unlikely that a current DSL or cable access network will support prioritization, so we propose to use probing to establish whether lower-priority streams are interfering with higher-priority ones. If the stream for the chunk that is being viewed is receiving insufficient bandwidth, the client can suspend the lower-priority streams and observe if that significantly increases the bandwidth of the high-priority stream. If so, this indicates that these streams are competing for the same bottleneck bandwidth and it is counterproductive to use multiple streams. The third approach is the creation of multiple TCP connections and retrieving the different chunks from different peer nodes. (Protocols such as RTSP allow random access within a file, so that random access is easily supported.) It should be noted that web browsers routinely create up to four simultaneous TCP connections to fetch both HTML and image content for a page, so that this behavior falls within standard Internet application practice.

5. SYSTEM DESIGN

For the architecture proposed in the previous section, this section gives the design considerations for providing the VCR operations with low seek latency, low startup time, good viewing experience by maintaining a minimum download rate and smart peer group selection for the nodes.

5.1 Minimum threshold downloading rate

All the nodes would be uploading to more than one node, so the uploading bandwidth would be split across multiple nodes. The sliding window will initially open a single connection and maintain a minimum downloading rate, which will be slightly more than video encoding/playback rate in order to prevent the video player from running into an empty buffer. When the downloading rate drop below the minimum threshold, the sliding window would open multiple connections and download the needed chunks, or else the pre-fetching module will open multiple connections and download chunks. The minimum downloading rate is very important to maintain a good viewing experience.

5.2 Anchor points

When the user does a VCR operation, and if the chunks for that location of the video have not been fetched, the user would experience a delay while the chunks are being fetched. So certain key/popular frames forming the 5 con-

tinuous seconds of video at regular intervals of 1 minute of video would form the anchor points [3], which would be downloaded during startup. Most of the user's forward seeks would also be to these key frames like the previous users, which could be served immediately using the chunks buffered in anchor points. If the user's seek is not to the exact anchor point, the sliding window would be adjusted to the nearest anchor point. Thus, the random seek is immediately satisfied without any delay and the 5 seconds of the playback time would be used to download the rest of the sliding window. These anchor points also helps in supporting the movie sampling by user when they scan through the video, much like the "intro-sampling" mode supported by portable CD-players. Results from experiments on user-behavior show that the most popular video content have smaller session lengths [12], where most of the users would have seen the most popular video before, either through another medium (theater or DVD) or in a prior VOD session. In most of these smaller sessions, the user's tend to just scan through the entire video, [12]. Downloading of anchors(key-frames) and most popular chunks, based on user behaviour would help in serving the chunks at a faster rate, when the user scans through the entire video file.

5.3 Size of sliding window

Size of sliding window is an important configuration parameter for maintaining of the minimum downloading rate, as the sliding window will be sliding forward at the rate of video encoding/playback. Based on experiments done in [4], 80% of the user's forward seeks are within the next 300 seconds of the video. So it would be ideal to have the sliding window size to buffer chunks of 300 seconds of the video. The sliding window will have to buffer all the chunks within the sliding window in the order of preference. Preference would be given for chunks approaching the chunk deadline the earliest. When the window is full, it will move one chunk forward every time a chunk of the video is consumed by the video player. When more than 90% of the window is empty, multiple connections would have to be opened and the minimum downloading rate should be maintained. This buffer underflow will usually happen when the user does a VCR operation to an unbuffered location.

Even when minimum downloading rate of the sliding window helps in real-time viewing, simulations in [6] show that due to pre-fetching and increasing the TTL (content buffering time, as defined in [6]) of a 90 minute movie file to 1000 minutes from 200 minutes, enabled the system to serve a certain number of nodes in the system at an uploading bandwidth rate of 1Mb/s. Earlier the same number of nodes could only be served with 4Mb/s of uploading rate with a TTL of 200mins. The pre-fetching module by buffering the content for a longer time helps to reduce the uploading bandwidth to one fourth the previous rate; this is a significant saving of the uploading bandwidth for content providers. The large storage capacities of the STBs help us in managing the large crowds by just increasing the buffering time i.e. TTL, for a constant uploading rate. This kind of cost savings on bandwidth usage and ease of load balancing cannot be achieved in content distribution network models.

6. SCHEDULING POLICIES

Scheduling policies define how to discover the available chunks, what chunks to download and where to download the chunks from. These policies will affect the overall performance of the entire VoD system as they define the amount of control messages being passed around the system and they also decide the efficiency and the timely delivery of the data received by the nodes. Section 6.1 describes how the global chunk index DHT of the movie, chunk map is setup and how the nodes query for the required content; Section 6.2 will identify which chunk should be given more priority while downloading and Section 6.3 will explain the locality-aware content fetching and its advantages.

6.1 Chunk Map

For every movie published by the VoD service, there will be one global DHT formed by all the nodes in the overlay that buffer the movie content. This is know as chunk map. Initially the entire movie would be stored in the source servers and all the chunks would have to be sourced from the source server. As the STBs download the chunks, they will update their position in the DHT with all the chunks available in them and remove the node from the DHT when the node leaves the network. One node in the overlay will be responsible for a specified range of chunk IDs of the video file in the DHT. This node will maintain a list of all the other nodes that have buffered the chunks of the IDs falling within its ID space. When every a STB want's to download a certain chunk, it will search through the DHT and get a list of all the nodes in the overlay which have buffered the required chunks.

6.2 Chunk selection policy

We can distinguish three types of chunks that a viewer can retrieve while watching a movie. The first chunk is the one currently being watched, which clearly has to have the highest priority to avoid disruptions in playback. This retrieval needs to proceed at least as fast as the consumption rate for the current chunk; if the download speed is larger, the content is buffered to compensate for any future glitches in delivery. The other types of retrievals make it more likely that VCR operations suffer minimum delay. The first type of prefetching obtains key frames for the whole movie, allowing smooth fast forward operation. For MPEG-encoded movies, this would typically be 1/15 of the frames of the movie, albeit a larger fraction of the byte volume. The lowest priority is given to prefetching chunks ahead of the current one. This can either be done sequentially, or the system can sample viewing histories to determine if a certain DVD chapter, for example, is far more popular than others. In-

stead of downloading a single segment (set of continuous chunks, forming a few minutes of the video) as a whole, we make use of the fact that viewers jump to the beginning of a chapter or scene. Thus, as in [3], we first download the first few chunks of each segment, enough to allow the system to stream the now-current segment without disruption. We repeat this striping for each of the segments. Since the three types of downloads are likely to come from different sources and download bandwidth is likely to be higher than upload bandwidth, the total available bandwidth for the viewer will be larger compared to pulling data from a single source. This policy also address the user observation [4] that users are more likely to skip ahead during later parts of a movie. At that time, more of the movie will be pre-cached at the viewer's STB. Once the nodes decide on the chunks to download, it will query the chunk-map or the global DHT for STBs that buffer these required chunks. The chunk-map returns a set of STBs buffering these chunks. The STB offering more uploading bandwidth among all the peers within the same subnet or AS (Autonomous System) would be selected for downloading of the chunks.

6.3 Locality-aware content fetching

In our architecture, the local DHT maps movie names to peer content identifiers as well as the server-based URL, should there be no available local (peer) copy or if all local STB holding copies are otherwise busy. Since each chunk is likely to be held by multiple STBs, a receiver needs to decide which STB to contact. In general, the selection process should avoid hot spots and favor short transmission distance between serving and downloading peer. Once the chunk comes over to a node within the subnet it will be exchanged locally, reducing the expensive cross *AS* traffic.

6.3.1 Content fetching by sliding window and pre-fetching modules

Sliding window would try to get the chunks needed by the sliding window. It will try to get the chunks sequentially before their *chunk deadline* expires. Sliding window module first searches for the required chunk from peers within the subnet, by searching through the DHT. If it is not available, then it will search for the chunk in other subnets by searching through the DHT. If the sliding window is not able to get the video chunk from within the overlay network, it will get the chunk from the source server. In order to maintain a fan out limit for each server and prevent excessive crowding at the source, there is a limitation that has been imposed on the maximum number of peers that can simultaneously try to fetch a chunk from a single node. The priority would be given to the sliding window connection, if it requests for a chunk over a pre-fetching connection. If the fan-out limit has reached for a particular node and a sliding window requests for a chunk, then application could preempt one of the pre-fetching connections and allow the sliding window connection to fetch the required chunk. The differentiation of the sliding-window and pre-fetching connections could be done based on the source port numbers. The pre-fetching client joins the same DHT as the sliding-window, and would get the same nodes for downloading of the chunks and both will download the chunk from the peer providing more upload bandwidth than the others.

6.3.2 Biased Neighbor selection advantages

According to [2], the main conclusion is that biased neighbor selections, in which a peer chooses the majority, but not all, of its neighbors from peers within the same subnet, can reduce expensive cross-subnet or cross *AS* traffic significantly. First, we want to encourage local retrival. We generally assume that there a broad equivalence classes of nodes, sharing the same switching equipment and possibly physical link. Based on existing DSL and cable architectures, it is likely that there will be three "rings" or proximitly, namely a single household, a subnet sharing the same router and an AS. In our design, nodes in the same subnet are identified easily using the common prefix of the IP addresses. When the location aware peer selection is adopted by the STBs, the chunk has to come over to the subnet only once, and then it will be duplicated within the subnet. Clearly, always choosing the first most-local peer from the list is likely to lead to admission control failures, so the STB could choose a random peer, excluding those that it is already using for retrieving other chunks.

It is not clear whether a more global optimization is necessary or helpful. For example, if chunks within the same movie differ dramatically in popularity, it could be advantageous to download those first, as it would increase the number of available copies. This, however, matters only for movies that have a very high number of simultaneous viewers, e.g., during their initial VOD release. During the first viewing, most viewers are likely to watch the whole movie, rather than select chunks. Yu et al. [12] have investigated user behavior, but not specifically distinguished between behavior for new movies and those that have been in circulation for a while.

Another global optimization would attempt to minimize the chance that a node holding rare content is busy serving content that is easily available from other, non-busy peers. Such differentiation is likely to matter only if the number of upstream sessions is very small, say, one or two, as otherwise the popularity of stream content stored on a node is likely to average out. Since this case should be rare and since a global optimization is infeasible, we propose instead to allow displacement. If a node has failed to gain admission to any peer with the desired chunk and if the chunk is the currently playing one, a peer sets a "last resort" flag and retries the candidate peers. Peers serving non-current chunks then remove one or more peers that are using that peer to prefetch chunks; these peers can retry elsewhere. The slidind-window connection connection would preempt other connections. Since 80% of the viewing is concentrated on

23% of the content [12], it is likely that the displaced peer is watching a popular movie and can readily find another!

7. PARAMETERS TO EVALUATE THE VOD SERVICE AND ADMISSION CONTROL

Startup latency is the time taken for the application to start playing of the video, once the user requests it. It includes the time to download first sliding window and all anchor points. Seek latency is the time taken to buffer the contents in a sliding window and start to playback, when the user does a VCR operation. These two are the parameters to evaluate the performance of the VoD system.

7.1 Admission control

Admission of any new incoming peer should be verified, so that the admission of the new peer will not lead to peformance degradation or 1001st stream problem: the admission of one stream results in a large number of subscribers experiencing degradation of the video signals, [11]. One way of avoiding a widespread perceived outage is through the use of integrated admission control. If a video session cannot be supported due to oversubscription anywhere in the network or service, this integrated control would deliver a "could not be serviced at that time" signal to the requesting set-top box. We support a distributed admission control in our model, where the admission of new node will depend on the extent to which the entire video content is distributed in the network and the number of source nodes available. When a new node requests for a movie, the client software will buffer the initial segments corresponding to the size of sliding window and the anchor points during the initial startup. If it is not able to meet the required minimum threshold downloading rate and the startup delay is exceeding the maximum permitted threshold duration, the p2p overlay is said to be crowded and the admission or new nodes should be avoided. This architecture has the provision of adding new source nodes at run time by updating the global DHT for the movie with new source nodes, when the content provider detects the degradation of the video signal to the existing users with addition of new subscribers to the movie.

8. CONCLUSION

Peer assisted VoD service delivery model is the key for streaming high definition video content, which has high bandwidth requirements. We have proposed a new design for providing VoD service using a peer-to-peer overlay network, which provides good viewing experience by maintaining a minimum downloading rate, at the same time doing a location aware content fetching and active pre-fetching of the popular chunks for better utilization of idle resources. This model also supports VCR operation better through anchors and also guarantees the QoS for the prescribed subscribers by controlling the admission of new nodes into the overlay. This model solves many problems, but it dosen't completely solve the problem of serving the non-popular content in overlay, without overloading the source servers. Future work could include the understanding of user behaviour, arrival pattern, fully integrated admission control and adapt to the preferences of the user, which would help in distributing the content in the overlay closer to the nodes, when there is low network traffic.

9. REFERENCES

[1] S. Annapureddy, C. Gkantsidis, and P. Rodriguez. "Providing video-on-demand using peer-to-peer networks". In *Internet Protocol TeleVision (IPTV) Workshop, WWW 06*, Edinburgh, Scotland, May 2006.

[2] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection,. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS 2006)*, Lisbon, Portugal.

[3] B. Cheng, H. Jin, and X. Liao. "RINDY: A Ring Based Overlay Network for Peer-to-Peer Streaming". In *3th IEEE Conference on Ubiquitous Intelligence and Computing (UIC06)*, Wuhan, China, Sep 2006.

[4] B. Cheng, X. Liu, and H. J. Zheng Zhang. "A Measurement Study of a Peer-to-Peer Video-on-Demand System". In *IPTPS*, Bellevue, Washington, USA, Feb 2007.

[5] B. Cohen. Incentives build robustness in bittorrent. In *First Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, June 2003.

[6] Y. Huang, Y.-F. Chen, R. Jana, H. Jiang, M. Rabinovich, B. Wei, and Z. Xiao. Capacity analysis of mediagrid: a p2p iptv platform for fiber to the node (fttn) networks. In *IEEE Journal on Selected Areas in Communication (JSAC), Peer-to-Peer Communications and Applications*, Jan 2007.

[7] S.Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. Rodriguez. "Exploring vod in p2p swarming systems". In *IEEE Infocom*, Anchorage , Alaska , USA, 2007.

[8] H. Schulzrinne, A. Rao, and R. Lanphier. "Real Time Streaming Protocol (RTSP) ". In *RFC - 2326*.

[9] P. Shah and J.-F. Pris. "Peer-to-Peer Multimedia Streaming Using BitTorrent". In *IPCCC 2007*, New Orleans, USA, April 2007.

[10] K. Suh, C. Diotz, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello. "PushtoPeer VideoonDemand system: design and evaluation". In *UMass Computer Science Techincal Report 200659*, Amherst, MA, USA, November 2006.

[11] C. systems. "Optimizing Video Transport in Your IP Triple Play Network". In *white paper from Cisco systems*, 2006.

[12] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. "Understanding User Behavior in Large-Scale Video-on-Demand System". In *EuroSys*, Leuven, Belgium, 2006.