P2P Live Streaming

Similar to many P2P file sharing or streaming systems, a P2P-VoD system has the following major components:

(a) a set of servers as the source of content (e.g., movies);

(b) a set of trackers to help peers connect to other peers to share the same content;

(c) a bootstrap server to help peers to find a suitable tracker (e.g. based on which geographical region the peer is located), and to perform other bootstrapping functions;

(d) other servers such as log servers for logging significant events for data measurement, and transit servers for helping peers behind NAT boxes.

These servers are typically provided by the P2P-VoD operator. The other major component, of course, is the set of peers. They typically run software downloaded from the P2P-VoD operator. The P2P-VoD peer software comes with protocols to talk to all the servers above, as well as protocols to talk to other peers to share content. The peers also implement DHT (distributed hash table) function to back up certain bootstrapping servers.

Segment Sizes

In the design of a P2P-VoD system, a fundamental decision is about segmentation of content, or how to divide a video into multiple pieces.

There are many considerations for making this decision.

From a scheduling point of view, it is desirable to divide the content into as many pieces as possible (i.e., small segment size), so that it gives the most flexibility to schedule which piece should be uploaded from which neighboring peer. This is specially so when peers all have different upload capacity.

From the overhead point of view, the larger the segment size the better, to minimize overheads. There are several types of overheads including: (a) Each piece of content comes with some header to describe the content, for example its sequence number and timestamp, and authentication information. The larger the segment, the smaller the header overhead. (b) Each peer needs to let other (neighboring) peers know which pieces it is holding. This information is usually represented by a bitmap, for ease of processing. The larger the segment size, the smaller the size of the bitmap, hence this is advertising overhead. (c) In order for a peer to get a piece of content from another peer, there will be some protocol overhead, in terms of request packets or other protocol packets. The larger the segment size the smaller is such protocol overheads.

A third perspective is due to the real-time nature of streaming. The video player expects a certain minimum size for a piece of content to be viewable (so a viewable piece always consists of multiple packets), and such viewable units must be delivered to the player with deadlines. By making these units (exchanged between the transport and the player) too large, it increases the chance that the transport fails to collect a complete viewable unit of content before the deadline.

PPLive Segment Sizes

Segment	Designed for	Size
movie	entire video	$>$ 100MB
chunk	unit for storage and advertisement	2MB
piece	unit for playback	16KB
sub-piece	unit for transmission	1KB

Replication Strategy

- Each peer is assumed to contribute a fixed amount of hard disc storage (e.g., 1GB). The entire viewer population thus forms a distributed P2P storage (or file) system.
- A chunk is the basic unit for storing movies on a hard disc. Only when all the pieces in a chunk are available locally, the chunk is advertised to other peers.
- The goal of the replication strategy is to make the chunks as available to the user population as possible to meet users' viewing demand while without additional overheads.

Replication issues

- Whether to allow multiple movies be cached if there is room on the hard disc. If so, a peer may be watching one movie while providing uploading to another movie at the same time. This is referred to as *multiple movie cache* (MVC) rather than *single movie cache* (SVC). The design of SVC is simpler, but MVC is more flexible for satisfying user demands and is the choice by PPLive VoD.
- The next important design consideration is whether to prefetch or not. Without prefetching, only those movies already viewed locally could possibly be found in a peer's disk cache. While Prefetching may improve performance, it may also unnecessarily waste precious peer uplink bandwidth. Also, for ADSL (commonly found in China), a peer's capacity to provide upload can be affected if there is simultaneous downloading. Furthermore, it is observed that the visit duration for the majority of peers is currently no more than one hour, which increases the risk of wastage. For these reasons, the design choice is no prefetching.
- Another important choice by the replication algorithm is which chunk/movie to remove when the disk cache is full. In PPLive's case, this decision is primarily made on a movie basis. This means once a movie has been chosen as the next one to go, all the chunks of the movie immediately become candidates for removal one by one. Doing it at a chunk level would incur more overheads (for collecting necessary information about different chunks). How is the next movie picked? The favorite choices by many caching algorithms are *least recently used* (LRU) or *least frequently used* (LFU). LRU is the choice in PPLive VoD. The simple LRU has been replaced by a weight-based evaluation process.

Weight based film evaluation

- Each movie is assigned a weight based on primarily two factors: (a) how complete the movie is already cached locally; (b) how needed a copy of the movie is.
- The *need* level is determined by the *availability to demand* ratio (ATD).
- Suppose a movie is cached (including being viewed) by *c* peers and being viewed by *n* peers; then the ATD is *c/n*.
- The *need* of a movie is defined as a decreasing function of its ATD, reaching a maximum value for all ATD beyond 6 (or 8). The value of this threshold (6-8) is determined by the prevailing uplink bandwidth contributed by peers, normalized by the source bitrate. Currently in China, many peers have relatively low uplink bandwidth to contribute, therefore it takes 6-8 peers to offload the source (server).
- The ATD information for weight computation is provided by the tracker. So the implementation of the weight-based replication strategy incurs additional overheads. This overhead depends on how often the caching decision is made. In current systems, the average interval between caching decisions is about 5 to 15 minutes, so this is not a significant overhead.
- The benefit of weight-based replication over LRU is significant. It improves the server loading from 19% down to 11% to 7%.

Content Discovery and Peer Overlay Management

- It is not enough to have good replication of content - peers must also be able to discover the content they need and which peers are holding that content. The challenge is to accomplish this with the minimum overhead.
- Without exception, P2P systems rely on the following methods for content advertising and look-up: (a) tracker (or super node); (b) DHT; (c) gossiping method. These methods provide different levels of availability, freshness and robustness, with corresponding levels of overhead. In PPLive VoD, all these mechanisms are used to some extent, depending on the different requirements for the information.
- Trackers are used to keep track of which peers replicate a given movie (or part of that movie). As soon as a user (peer) starts watching a movie, the peer informs its tracker that it is replicating that movie; conversely, a peer also tells its tracker when it no longer holds a movie in its cache.
- When a peer wants to start watching a movie, it goes to the tracker to find out which other peers have that movie. Those other peers become this peer's *neighbors*.
- The information about which chunks a peer has is kept in a *Chunk Bitmap*. A peer asks its neighbors for their Chunk Bitmaps. Based on this information, it selects which neighbor to download from. So discovering where chunks are is by the *gossip* method. This cuts down on the reliance on the tracker, and makes the system more robust. Even if the tracker is not available, a peer can switch to the gossip mode to find other peers watching the same movie.
- In fact, each peer also regularly sends keep-alive messages to a tracker to report its chunk bitmap and other statistics. This information is collected for monitoring and management purposes, rather than for operational reasons. This information is used to compute a (replication) *health index*.
- Originally, DHT (implemented by tracker nodes) is used to automatically assign movies to trackers to achieve some level of load balancing. In later versions, peers also implement DHT so as to provide a non-deterministic path to the trackers. This prevents the trackers to be possibly blocked by some ISPs.

Piece Selection

A peer downloads chunks from other peers using a pull method.

For P2P-VoD, there are three considerations for selecting which piece to download first:

1. sequential: Select the piece that is closest to what is needed for the video playback.
2. rarest first: Select the piece that is the rarest (usually the newest piece in the system). Although it seems counter-intuitive for streaming, selecting the rarest piece helps speeding up the spread of pieces, hence indirectly helps streaming quality. This strategy tends to help the system scale.
3. anchor-based: In VoD, users may skip parts of a movie and jump forward (backward). To support such VCR features, a number of video anchor points are defined for a movie. When a user tries to jump to a particular location in the movie, if the piece for that location is missing then the closest anchor point is used instead.

In PPLive's system, a mixed strategy is used, giving the first priority to sequential, then rarest-first. The anchor-based method is not used in current design for two reasons. (a) From current experience, users do not jump around much. On average, only 1.8 times per movie is observed. (b) By optimizing the transmission scheduling algorithm, the initial buffering time after a jump can be reduced to an acceptable level without anchoring.

Transmission Strategy

- After selecting a particular chunk to download, suppose this chunk is available at a number of neighbor peers, how to select which neighbor to download from?
- How many neighbors to use for simultaneous download?
- How to schedule requests and set timeouts to multiple neighbors for simultaneous download?

All these are accomplished by the transmission scheduling algorithm.

There are two (sometimes conflicting) goals in designing the transmission algorithm: (a) maximize (to achieve the needed) downloading rate; (b) minimize the overheads, due to duplicate transmissions and requests. In a data-driven overlay, the neighbors a peer connects to can be highly dynamic, since each neighbor may be answering to multiple requests at a time.

A peer must constantly try how to send download requests to different neighbors and how to deal with timeouts.

There are different levels of aggressiveness:

(i) a peer can send a request for the same content to multiple neighbors simultaneously, to ensure it gets the content in time;

(ii) a peer can request for different content from multiple neighbors simultaneously; when a request times out, it is redirected to a different neighbor;

(iii) work with one neighbor at a time; only when that neighbor times out, try to connect to a different neighbor.

Strategy (i) is very aggressive for achieving the deadline for downloads, but invariably generates duplicate transmissions. Strategy (iii) is very conservative in resource utilization. Both strategy (ii) and (iii) may still general duplicate transmissions because of timeouts, but the likelihood is much lower than (i).

PPLive VoD's transmission algorithm is based on strategy (ii). In implementing strategy (ii), the algorithm tries to proportionally send more requests to the neighbor based on response time. A critical parameter for tuning is the number of simultaneous neighbors to send requests to. For playback rate of around 500Kbps, experience shows that 8-20 neighbors is the right number. More than this can still improve the achieved rate, but at the expense of heavy duplication rate. If the desired rate is 1Mbps,then 16-32 simultaneous neighbors tends to provide the best result. These numbers are highly empirical, depending on the prevailing uplink bandwidth of peers and many other factors. Overall, how to design the best transmission algorithm is an interesting topic for further research. Finally, it should be pointed out that when the neighboring peers cannot supply sufficient downloading rate, the content server can always be used to supplement the need.