Signal Processing: *Image Communication* 27 (2012) 484–495

Contents lists available at SciVerse ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

Server-assisted adaptive video replication for P2P VoD

Yipeng Zhou, Tom Z.J. Fu, Dah Ming Chiu*

Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong

ARTICLE INFO

Available online 16 February 2012

Keywords:
Peer-to-Peer
Video-on-Demand
Replication

ABSTRACT

In recent years, Peer-to-Peer assisted Video-on-Demand (P2P VoD) has become an effective and efficient approach to distribute high-quality videos to large number of peers. In a P2P VoD system, each peer contributes storage to store several videos to help offload the server. The replication strategy, which determines the videos to be stored at each peer's local storage, plays an important role in system performance. There are two approaches: (a) solve a huge combinatorial optimization problem and (b) use simple cache replacement algorithms, such as Least-Frequently-Requested (LFR) or FIFO. The first approach needs to collect a large number of parameters whose values may be changing, and use some approximation method (such as linearization) to solve the optimization problem, both aspects have accuracy issues. In the second approach, a peer replaces some video in the cache with the currently viewed video, based on local information. While it is simple, we show their performance can be improved by a little centrally collected state information. Specifically, the needed feedback information is the current downloading rate provided by peers for each video. In this paper, we describe a hybrid replication strategy, and give detailed description of how the server collects and maintains the feedback information, and how peers use that information to determine what videos to store and indirectly control their uplink bandwidth contribution. This explains why the hybrid strategy is much simpler and more practical than the combinatory optimization approach. We then use simulation to demonstrate how our scheme out-performs the simple adaptive algorithms. Our simulation results also demonstrate how our scheme is able to quickly respond to peer churn and video popularity churn.

1. Introduction

Traditional Video-on-Demand (VoD) is based on the client-server approach. It is expensive and not scalable. In recent years, the Peer-to-Peer approach was first demonstrated to work for large-scale live content streaming [1], and later for large-scale VoD streaming as well [2]. Various efforts are now trying to build a P2P-based VoD platform, for example using set-top boxes [3]. This framework can be used to support high quality streaming and high definition videos.

Realizing a P2P VoD streaming service is more challenging than P2P live streaming. In P2P live streaming, peers watching the same broadcast naturally have (parts of) the same content to share. In P2P VoD, peers are less likely to have the same content to share with each other. The lack of synchrony (in VoD) is compensated by the following two measures: (a) each peer is capable of uploading content different than what is currently consumed (downloaded) locally and (b) peers contribute additional storage to replicate content (for uploading when it is not being viewed locally). The effectiveness of these measures depends on the content placed at different peers, which is

* Correspondence to: Room 836, SHB, CUHK, Hong Kong. Tel.: +852 2609 8357.
E-mail address: dmchiu@ie.cuhk.edu.hk (D.M. Chiu).

0923-5965/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.image.2012.02.010

the P2P replication problem at hand. As discussed in [2], P2P replication is a central design issue in P2P VoD systems.

The objective of replication strategy is to minimize the server load and satisfy users' streaming requirement at the same time. The streaming requirement means there needs to be a balance between the total supply of uplink bandwidth (that is the sum of server(s) and peers' uplink bandwidth) and the total demand (that is the number of viewing peers multiplied by the video playback rate). In practice, the operating regime of particular interest is when the total peer uplink bandwidth is comparable to the demand (of viewing bandwidth). In this regime, there is the greatest potential for the intelligent placement of content in peers to offload the server. Ideally, the server bandwidth used can be negligibly small, if the viewing demand is deterministic and known a priori, and all the peers replicate sufficient content so as to make full use of their upload capacities. However, in reality, the unpredictability of user demand, and hence the imperfection in content replication and service load balancing will always result in some server load [4].

The P2P replication problem has been studied before, for different service models (P2P search, file sharing, downloading and streaming VoD). For current P2P VoD systems, there are basically two solutions: (a) optimization-based [4–7] or (b) *lazy* adaptive. Approach (a) usually involves large-scale information collection, and dealing with an NP-hard computational problem. Even after simplification and approximation, the solution still takes a non-trivial amount of resources, and can only be afforded once in a while (e.g. once a day or once a week) [5]. At this scale, the parameters collected cannot be very accurate. Approach (b) usually means some simple caching schemes such as Least-Frequently-Requested (LFR) or Random (Rnd). The schemes are indeed quite easy to implement, and highly adaptive to loading. But can we do better?

In this paper, we take a hybrid approach, and consider the P2P replication problem as an adaptive control problem. A server helps collect key performance indicators that are used by peers to adjust the content they replicate. Two kinds of information are collected in each time session: first, the number of existing copies for each video; second the downloading rate (from other peers) for each video.[1] This approach allows us to deal with peer churn, channel churn (which happens more than peer churn [8]), and even non-stationary popularity of movies. Optimality and responsiveness can be traded off with system overheads.

Simulation is used to validate our results and compare our solution with some other distributed replication strategies (LFR, FIFO and Random) under stationary, peer churn and movie popularity churn cases. We show that our replication strategy performs significantly better than the other strategies.

In the rest of the paper, we first describe our model in Section 2, the replication strategy in Section 3, and derive our analytical result in Section 4, and show our simulation results in Section 5. Before we conclude the paper, we also discuss related works and explain the significance of our work in light of the earlier publications.

2. System model and assumptions

Our model assumes there are N peers and M movies. Each peer contributes storage space enough to store L movies, where $L \ll M$. There is a (or more than one) server that stores all the movies and serves as a backup whenever a peer cannot achieve required downloading rate (equal to playback rate). Furthermore, all peers are reachable from each other, and the only bottlenecks in the network are the peers' uplink bandwidth. To complete the description of the system, we need to describe (a) a movie request model, (b) a service scheduling model, and finally (c) an adaptive control model.

For simplicity, we assume all movies are of the same size and with the same playback rate $R_j=1$, for $j=1,2,\ldots,M$. The movie request model is based on a queueing network model as described in [8]. Basically, the time duration for each peer to stay with any movie follows an exponential distribution, which is introduced in work [2,17]. Then, the peer switches to other movies based on a transition matrix. The consequence is that the number of requests for each movie follows a Multinomial distribution with parameter N and $(\eta_1,\eta_2,\ldots,\eta_M)$ with $\sum_{j=1}^{M}\eta_j=1$. We use the request vector $(n_1,n_2,\ldots,n_M)$ to represent the expected number of requests for each movie at any moment and $n_j=N\times\eta_j$. This is our *request model*[2].

To satisfy each peer request for a movie, we assume the P2P system provides a *fair sharing* service model [4]. In other words, a request for movie j is sent to all peers storing that movie; and each peer lets all requests share its uplink bandwidth equally. Actually, the fair sharing model is a fluid model that allows the uplink bandwidth to be divided into any small value. An immediate consequence is that the peers selecting the same video will receive the same bandwidth resource, hence the same downloading rate. This is our *service scheduling model*. Although this is an abstraction, it is approximated by many practical P2P content distribution systems, for example the large scape P2P streaming system described in [2].

Finally, we can describe our adaptive control model. The state of the system is Q_i, the set of movies stored at a peer i, for $i=1,\ldots,N$. From this, we can compute $(c_1,c_2,\ldots,c_M)$, the number of peers replicating movie $1,2,\ldots,M$. To simplify the control mechanism, we make a deliberate decision to not control Q_i, exactly which movies are stored at each peer i, but c_j, the number of

[1] Since we are using a fair sharing model to serve peers, all peers downloading the same movie get approximately the same downloading rate. The detailed scheduling policy for the fair sharing model is explained later.

[2] Actually, as will be evident from our simulation model, all we assume is that movie requests follow a Multinomial distribution. The queueing network formulation from [8] helps visualize this popularity model, but is not necessary. Later on, we will assume the resulting vector η conform to a Zipfian distribution, for ease of analysis and presentation.