Computer Networks 54 (2010) 1229–1241

Contents lists available at ScienceDirect

Computer Networks

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

Proxy caching for peer-to-peer live streaming ☆

Ke Xu [a], Ming Zhang [b,*], Jiangchuan Liu [c], Zhijing Qin [d], Mingjiang Ye [a]

[a] Department of Computer Science, Tsinghua University, Beijing 100084, PR China
[b] School of Software, Tsinghua University, Beijing 100084, PR China
[c] School of Computing Science, Simon Fraser University, Vancouver, BC, Canada
[d] School of Software and Microelectronics, Peking University, Beijing 100871, PR China

ARTICLE INFO

Article history:
Received 6 May 2009
Received in revised form 25 September 2009
Accepted 29 November 2009
Available online 6 December 2009
Responsible Editor: I. Habib

Keywords:
P2P
Live streaming
Proxy caching

ABSTRACT

Peer-to-Peer (P2P) live streaming has become increasingly popular over the Internet. To alleviate the inter-ISP traffic load and to minimize the access latency, proxy caching has been widely suggested for P2P applications. In this paper, we carry out an extensive measurement study on the properties of P2P live streaming data requests. Our measurement demonstrates that the P2P living streaming traffic exhibits strong localities that could be explored by caching. This is particularly noticeable for the temporal locality, which is often much weaker in the conventional P2P file sharing applications. Our results further suggest that the request time of the same data piece from different peers exhibits a generalized extreme value distribution. We then propose a novel sliding window (SLW)-based caching algorithm, which predicts and caches popular data pieces according to the measured distribution. Our experimental results suggest that the P2P live streaming can greatly benefit from the proxy caching. And, with much lower overhead, our SLW algorithm works closer to an off-line optimal algorithm that holds the complete knowledge of future requests.

1. Introduction

Recent years have witnessed the tremendous success of the peer-to-peer (P2P) communication paradigm. With each participating node contributing its own resources, the P2P communication architecture scales extremely well with user population. It has been widely used in such applications as file sharing [1,2], voice over IP (VoIP) [3], live streaming and video-on-demand (VOD) [4]. They together have contributed to a great portion of the overall Internet traffic [5], and its ever growing trend has posed a significant threat to sustainable operations of Internet Service Providers (ISP) [6].

To mitigate the traffic load, particularly the costly inter-ISP traffic, caching data of interest closer to end-users has been frequently suggested in the literature. There have been extensive studies on caching the traffic of web [7,8] or streaming video in the client/server architecture (video streaming for short) [9]. Recent works have also examined caching for P2P file sharing [6,10]. The caches are generally deployed at gateways of institutions, referred to as *proxy caching*. Through satisfying requests from the local storage, they not only reduces the bandwidth consumption, but also minimizes the access latency [11]. The latest experiments further suggests that caching is very effective for P2P file sharing, too (with a bandwidth reduction of over 60% [12]).

There are however important differences between living streaming and the conventional web and file sharing applications. For the latter two, the data pieces exhibit a skewed distribution, and the pieces frequently requested

☆ This work has been supported in part by NSFC Project (60970104, 60873253), NSFC-RGC Joint Research Project (20731160014), 973 Project of China (2009CB320501), 863 Project of China (2008AA01A326) and Program for New Century Excellent Talents in University.

* Corresponding author. Tel.: +86 6 260 3064; fax: +86 6 277 0753.
E-mail addresses: xuke@tsinghua.edu.cn (K. Xu), tigerCN7@gmail.com (M. Zhang), jcliu@cs.sfu.ca (J. Liu), qzj@csnet1.cs.tsinghua.edu.cn (Z. Qin), yemingjiang@csnet1.cs.tsinghua.edu.cn (M. Ye).

1389-1286/$ - see front matter
doi:10.1016/j.comnet.2009.11.013

in the past tend to be frequently requested in the future. This is not the case in living streaming that demands sequential playback. The data pieces in living streaming on the other hand has strong temporal correlations, since they are exchanged in a small playback window. With the playback window moves over time, the popularity and even usefulness of the data pieces change as well. To the best of our knowledge, proxy caching for P2P live streaming traffic has yet to be examined, despite several pioneer studies on collaborative caching on the peer side [13,14].

In this paper, we present a systematic study on the proxy caching for P2P live streaming. We first analyze the real data request of a popular P2P live streaming application, PPLive [4], and identify its key characteristics. In particular, we find that the request time of the same data piece from different peers exhibits a generalized extreme value distribution. We then develop a data request generator that can closely synthesize P2P live streaming traffic. We further propose a novel sliding window (SLW) caching algorithm that explores the unique distribution of the live streaming requests.

We evaluated the performance of our algorithm and compare it with typical caching strategies. Our experiments show that our algorithm well suites the P2P live streaming traffic, and its cache hit rate is very close to that of an off-line optimal algorithm. We further model the theoretical performance of the new algorithm, and provides general guidelines for the configuration and optimization of the algorithm.

The rest of the paper is organized as follows. In Section 2, we survey the related work. In Section 3, we compare the traffic characteristics of P2P live streaming and P2P file sharing, followed by a measurement study on the data request distribution of PPLive in Section 4. We then present our caching algorithm in Section 5, and evaluate its performance in Section 6. Section 6 further models the performance of our caching algorithm. Finally, Section 7 concludes the paper.

2. Related work

There have been a series of measurement studies on P2P live streaming systems, particularly PPLive [15,16]. Hei et al. [15] carry out an in-depth analysis of PPLive, and find that PPLive users experience large start-up delay and playback lags. They conclude that dedicated proxy nodes are necessary to help with delivering videos at higher playback rate. According to Ali et al. [16], P2P live streaming has a greater impact on network bandwidth than P2P file sharing, because the upload bandwidth is unfairly exhausted by some peers. These studies reveal a lot of statistical information about PPLive, but the request distribution, which is of critical importance to cache design, has yet to be identified.

In the past, web caching has been extensively studied [8]. The importance and feasibility of caching P2P traffic have been demonstrated in [17,12] and [18]. Studies [17] and [12] show that P2P traffic is highly redundant and caching can reduce as much as 50–60% of the traffic. The work in [18] further suggests deploying proxy caches and making P2P protocols locality-aware, so as to reduce inter-ISP traffic. Other studies on proxy caching for P2P file sharing include [6] and [10]. We summary the key issues addressed in these studies as follows:

- *Object popularity.* Incorporating object popularity into caching algorithms can help with improving the cache hit rate. The pattern of object popularity is usually studied as part of the caching algorithm design process [6]. Typical object popularity patterns can be found in [7] and [19]. These studies all prove the effectiveness of the Pareto principle (or 80–20 rule). The channel popularity in P2P live streaming also satisfies the 80–20 rule which is an important consideration for caching algorithm design in our work.
- *Temporal and spatial locality.* Locality of reference characterizes the ability to predict future accesses to objects from the history information [8]. There are two important types of locality: temporal and spatial. Temporal locality refers to the repeated accesses to the same object within short time periods. It implies that recently accessed objects are likely to be accessed again in the future. Spatial locality refers to the patterns that accesses to some objects imply accesses to certain other objects. That said, the references to some objects can be a predictor of future references to other objects. Least Recently Used (LRU) is a classic algorithm utilizing temporal locality. The SLW algorithm proposed in our work exploits both the temporal and spatial locality of data requests.

Despite the common issues, existing studies have suggested that web, video streaming and P2P file sharing traffic all have their distinct features, and specialized policies are preferred to cache their respective traffic [10]. Also note that the popular pre-fetching policies to minimize the startup latency or to meet certain time constraints [9,20–23] are not necessarily useful for living streaming. This is because the data pieces are released gradually over time. In this work, we concentrate on achieving best cache hit rate without considering time constraint explicitly. We will see later that these two goals actually do not conflict.

3. Proxy caching for P2P live streaming: a general view

In this section, we first give a generic architectural view of caching for P2P live streaming, in particular, where and how the caches are deployed. We then analyze the differences between live streaming and file sharing in terms of cache design, which motivates our study.

3.1. Generic cache architecture for P2P traffic

In the P2P communication paradigm, each node (peer) has dual roles: downloading from other peers as a client, and uploading to other peers as a server. To avoid long-distance sessions, particularly those between peers of different institutions such as Autonomous Systems (ASes), the cache servers can be deployed at the access points of regional networks (as in [24]), as shown in Fig. 1a.

Once operated, the gateway intercepts the P2P downloading requests originated from its associated regional network and redirects them to the P2P cache server