Computer Networks 93 (2015) 199–212

Contents lists available at ScienceDirect

Computer Networks

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

Maximizing real-time streaming services based on a multi-servers networking framework

Tian Wang [a,*], Yiqiao Cai [a], Weijia Jia [b], Sheng Wen [c], Guojun Wang [d,e], Hui Tian [a], Yonghong Chen [a], Bineng Zhong [a]

[a] College of Computer Science and Technology, Huaqiao University, 668 Jimei Avenue, Xiamen, Fujian Province 361021, China
[b] Department of Computer Science and Engineering, Shanghai Jiaotong University, 800 Dongchuan Road, Minhang District, Shanghai 200240, China
[c] School of Information Technology, Deakin University, Victoria, Australia
[d] School of Computer Science and Educational Software, Guangzhou University, Guangzhou 510006, China
[e] School of Information Science and Engineering, Central South University, Changsha 410083, China

ARTICLE INFO

Article history:
Received 13 July 2015
Revised 15 October 2015
Accepted 26 October 2015
Available online 30 October 2015

Keywords:
Streaming
Multi-servers
Services
Real-time
Backup

ABSTRACT

In recent years, we have witnessed substantial exploitation of real-time streaming applications, such as video surveillance system on road crosses of a city. So far, real world applications mainly rely on the traditional well-known client–server and peer-to-peer schemes as the fundamental mechanism for communication. However, due to the limited resources on each terminal device in the applications, these two schemes cannot well leverage the processing capability between the source and destination of the video traffic, which leads to limited streaming services. For this reason, many QoS sensitive application cannot be supported in the real world. In this paper, we are motivated to address this problem by proposing a novel multi-server based framework. In this framework, multiple servers collaborate with each other to form a virtual server (also called cloud-server), and provide high-quality services such as real-time streams delivery and storage. Based on this framework, we further introduce a $(1-\epsilon)$ approximation algorithm to solve the NP-complete "maximum services" (MS) problem with the intention of handling large number of streaming flows originated by networks and maximizing the total number of services. Moreover, in order to backup the streaming data for later retrieval, based on the framework, an algorithm is proposed to implement backups and maximize streaming flows simultaneously. We conduct a series of experiments based on simulations to evaluate the performance of the newly proposed framework. We also compare our scheme to several traditional solutions. The results suggest that our proposed scheme significantly outperforms the traditional solutions.

1. Introduction

In recent years, we have witnessed the popularity of real-time streaming applications in the various environments, such as the video surveillance systems [1–3]. Not only can the police monitor the risky streets and road crosses, but also general public can employ these applications to enhance security and safety in their business and life. In these applications, there will be many cameras installed in order to cover as large area as one can. For example, Nasution and Foroughi et al. proposed methods to detect and record various posture-based events of interest in elder monitoring application at various home surveillance scenarios [4–6]. In particular, due to the popularity of smart phones and wide accessibility of

* Corresponding author. Tel.:+86 18359242669.
E-mail address: wsnman@gmail.com (T. Wang).

http://dx.doi.org/10.1016/j.comnet.2015.10.019
1389-1286/

wireless connections, people are able to use portable devices to attain the events of interest from remote cameras [7].

Due to the limited resources on each terminal device in the applications, current schemes of the real-time streaming applications cannot well leverage the processing capability and storage space between the source and destination of the video traffic. For this reason, many QoS sensitive application cannot be supported in the real world. For example, there are traditionally mainly two general architectures for real-time services. The first one is the client–server (C/S) scheme where the terminals stream the data captured in the places of interest to the rendezvous server. Remote users can then retrieve the data from the server. This scheme is simple but incurs heavy burdens on the server. It also does not scale well when more terminals of monitoring get involved [8]. Another widely-adopted scheme is the peer-to-peer scheme (P2P). The P2P scheme greatly alleviates the burdens on rendezvous servers, and improves the applications scalability [8,9]. For example, Pudlewski [10] proposed a point-to-point real-time video transmission scheme over the Internet in conjunction with a compression method that is error resilient and bandwidth-scalable. However, any terminal in the P2P scheme acts as both client and server. This leads to the heavy burden on the transmission source. Moreover, because the lightweight server (every terminal) is in short of storage space and processing capability, the P2P scheme cannot support streaming data retrieval. The situation becomes worse when the connections are not stable. The quality of service (QoS) will become unacceptable. To date, It has long been a big challenge to realize the real-time streaming over best-effort packet networks. On the other side, some real-time streaming data is required to be backed up for later retrieval. For example, the data produced by video surveillance system is usually required to be backed up for a long time, which can be retrieved for taking evidence.

In this paper, we proposed a novel framework based on multi-server to address the above problem. In this framework, we take the real-time data streaming and storage into account at the same time, and multiple servers will incorporate with a virtual server (also called cloud-server) to provide both forwarding and storage services. Since our newly proposed framework can fully utilize the resources from dynamic networks [11], the multiple servers provide better robustness in case one transmission channel becomes congested, and achieve high throughput on end users. We carried out a series of experiments in order to test and evaluate the performance of our proposed framework. Compared to the traditional single server scheme, the best server and route/path could be picked out in our solution. For the perspective of users, the service will be provided by a super server, and the multiple servers will coordinate with each other to maximize the simultaneous services.

We summarize the contributions of this paper as follows:

- We proposed a novel framework based on multi-servers to collaboratively provide services for each end user. This framework exploits the diversity of network and efficiently realize the real-time streaming and backup at the same time.
- Based on the proposed framework, we innovatively formulated a brand-new problem in providing real-time streaming services: Maximum Services (MS). This problem aims to maximize the total number of streaming flows and/or services. We proved that this problem is actually an NP-complete problem.
- For the real working applications of real-time streaming services, we innovatively inserted a two-relay restriction into the framework, and therefore simplified the MS problem. We designed an approximate algorithm with foreseeable performance bounds to the optimal solution. Extensive simulations were carried out to validate the effectiveness of the proposed methods for real working applications.
- A new metric Possible Loss (PL) was introduced to evaluate the storage and backup performance. An efficient storage algorithm is designed to achieve the minimized PL.

The rest of this paper is organized as follows: Section 2 presents related work. The details of the framework are presented in Section 3. The approximate algorithm of MS problem is presented in Section 4, followed by Section 5, which is about the performance evaluation. Section 6 concludes this paper.

2. Related work

Clearly, the amount of data in our industry and the world is exploding. Data is being collected and stored at unprecedented rates. For example, video streaming has become a popular application. In 2010, the number of video streams increased 38.8% to 24.92 billion even without counting the user generated videos [12]. Video news clips, movies and TV shows, and videos made and shared by the general public are watched by millions of people every day [13,14]. A considerable amount of data produced by the public needs to be delivered in real-time such as online chatting, online games, video surveillance and so on [15,16]. The challenge is not only to store and manage the vast volume of data, but also to deliver the “Big Data” in real-time.

Traditional client–server based data streaming solutions incur expensive capacity provision cost on the server and do not scale well [8]. On the other hand, peer-to-peer (P2P) streaming greatly alleviates the burdens on servers with good scalability [8,9] where users act as both clients and servers, which then incurs burden for data sources. Many people consider them as suitable infrastructures for supporting real-time streaming. However, P2P networks possess dynamic characteristics that can decrease drastically the performance of these real-time applications. If the communication link between peers becomes bad, the QoS cannot be maintained. In contrast to traditional work, we combine multiple servers into a cloud-server to collaboratively provide reliable and efficient services.

The authors in [17,18] also utilized “multiple servers” to stream media over a lossy packet network . In their work, media packets are typically characterized by different deadlines, importance and interdependencies. Using this information, the client is able to request the transmission of media packets from the multiple servers. They proposed a client–driven rate-distortion optimal packet scheduling algorithm that decides which packet(s) are to be requested from which server(s) at a given request opportunity. The difference between their work and ours is, we do not stream one data