Computer Networks 100 (2016) 75–89

Contents lists available at ScienceDirect

Computer Networks

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

An architecture for client virtualization: A case study

Syed Arefinul Haque[a], Salekul Islam[a,*], Md. Jahidul Islam[a], Jean-Charles Grégoire[b]

[a] United International University, Dhaka, Bangladesh
[b] INRS-EMT, Montréal, Canada

ARTICLE INFO

Article history:
Received 30 November 2015
Revised 17 February 2016
Accepted 18 February 2016
Available online 26 February 2016

Keywords:
Edge cloud
P2P
BitTorrent
Virtual client
Cloud-based server
Web-RTC

ABSTRACT

As edge clouds become more widespread, it is important to study their impact on traditional application architectures, most importantly the separation of the data and control planes of traditional clients. We explore such impact using the virtualization of a Peer-to-Peer (P2P) client as a case study. In this model, an end user accesses and controls the virtual P2P client application using a web browser and all P2P application-related control messages originate and terminate from the virtual P2P client deployed inside the remote server. The web browser running on the user device only manages download and upload of the P2P data packets. BitTorrent, as it is the most widely deployed P2P platform, is used to validate the feasibility and study the performance of our approach. We introduce a prototype that has been deployed in public cloud infrastructures. We present simulation results which show clear improvements in the use of user resources. Based on this experience we derive lessons on the challenges and benefits from such edge cloud-based deployments.

1. Introduction

A new trend in service deployment in the Internet, based on cloud computing and virtualization, shifts the location of applications and infrastructures from the user device to the network to reduce the costs associated with the management of hardware and software resources [1]. In such systems, service providers can provide simplified software installation, maintenance and update [2]. As cloud technology has become more popular, we have seen the emergence of edge clouds [3], that is, datacenters deployed by Internet access providers, in close proximity to customers. With such facilities comes new opportunities to shift traditional computer based applications, which could not otherwise have easily been virtualized because of their complex control plane, towards the cloud. Peer-to-peer (P2P) applications would be an example of such applications.

P2P networks are popular tools for content-sharing because they provide better scalability and fault tolerance than the traditional client-server model of computing. A P2P network can be described as a network of cooperating peers that work together to complete tasks and share resources in the Internet. Such a network is composed of numerous distributed, heterogeneous, autonomous, and highly dynamic peers with which participants share a part of their own resources such as processing power, storage capacity, software, and content [4]. P2P networks have no single point of failure and the network can grow and shrink without sacrificing the functionality of the system. Bandwidth utilization is better in such networks as the peers communicate directly with each other rather than through a hub which would present a bottleneck [5].

P2P applications are often used for file sharing. One example of a popular P2P file sharing application is

* Corresponding author. Tel.: +880 1820182777.
E-mail addresses: arefin@cse.uiu.ac.bd (S.A. Haque), salekul@cse.uiu.ac.bd (S. Islam), jahid@cse.uiu.ac.bd (Md.J. Islam), gregoire@emt.inrs.ca (J.-C. Grégoire).

http://dx.doi.org/10.1016/j.comnet.2016.02.020
1389-1286/© 2016 Elsevier B.V. All rights reserved.

BitTorrent. Large corporations like the Blizzard Inc. use P2P systems to simultaneously distribute bandwidth-intensive content to thousands of users without requiring major infrastructure investments [6]. On-demand media streaming is another popular P2P application. PPTV [7] delivers video content by streaming but peers can watch and share different parts of a video at the same time thus reducing server load [8]. Distributed P2P file storage systems like Freenet [9] anonymously publish, replicate and retrieve data distributed among peers. Skype [10] is a P2P application which enables voice and video calls over the Internet to any other Skype user. P2P model can even be used to virtualize physical objects and service construction processes on smart spaces [11]. Several distributed scientific projects like Seti@Home [12] use P2P public distributed computing to share processing cycles. BitTorrent Sync [13] is a recent addition to this paradigm that synchronizes files between devices on a local network, or between remote devices over the Internet.

For running such P2P applications a user normally has to install a client application on her device. The single most important task of these applications is to exchange data between peers, but apart from that, they may also perform routing/forwarding, content validation (e.g. hash checking) and implement different mechanisms for efficient bandwidth usage. As a result these client applications consume various resources including processing power, memory and bandwidth. Also, NAT traversal is an issue in P2P applications as most of the peers usually do not have globally routable IP addresses [14]. A local application requires a prior installation and has to be regularly updated for maintenance, which can be a burden for the user.

In this paper we explore how P2P applications benefit from virtualization in an edge cloud environment and study the architectural tradeoffs. For this process, we introduce *SimpleBit*, a virtual terminal-based P2P client which follows the BitTorrent protocol but is deployed on a remote cloud server. This architecture was first introduced in [15], albeit very briefly. An end user accesses and controls SimpleBit using a standard web browser, which reduces the requirements and the load on user devices by offloading the control and session management tasks to the remote server. We study two different architectures of SimpleBit:

1. A P2P-type direct download architecture where the files are downloaded directly from the peers to the end user's device.
2. A surrogate-based proxy downloader where the files are first downloaded by the surrogate server and then transferred to the end user's device.

The rest of the paper is organized as follows: Section 2 briefly describes BitTorrent and discusses a detailed overview of how the BitTorrent client works. Section 3 introduces and discusses virtualization. In Section 4 we present the architecture of SimpleBit virtual P2P client. We have designed two orthogonal models: SimpleBit with proxy downloader is explained with its implementation in Section 5 and SimpleBit with P2P download is presented with simulation results in Section 6. In Section 7 we explain the lessons we have learnt on the challenges and benefits from the edge cloud-based deployments. In Section 8 we summarize and compare existing efforts that are related to our work. Finally, Section 9 concludes the paper.

2. Dissecting BitTorrent

BitTorrent is the most popular P2P application for distributing large size files. It is implemented as a hybrid P2P system. Most of the interactions are done directly between peers but initial and further occasional interactions with a server are required for locating peers [16]. A user gets the information about the peers using a meta-information (metainfo) file (or metafile). The architecture of BitTorrent is shown in Fig. 1. It can be summarized in the following points:

1. A peer willing to download a shared content has to download the corresponding metafile from a web server and uses it to identify a *tracker* for that content.
2. The peer contacts the tracker and requests a list of peers that are already participating in the torrent (i.e., sharing that content).
3. The tracker replies with a list of peers with their IP address and access port.
4. The peer selects a number of peers from the list provided by the tracker and establishes a connection with them.
5. When connections are established, the peer exchanges pieces of that file with the neighbors.

A set of peers using the same metafile to share a particular file are part of the same *swarm*. A tracker can introduce the newly joined peer to multiple swarms at the same time. A file is divided into fixed-size pieces and peers exchange the pieces with each other. When a piece is downloaded its SHA1 hash is computed and compared with the value in the metafile. If the values match then the piece is declared downloaded and made available for downloading to other peers.

BitTorrent uses pipelining to keep the TCP connections operating at full capacity [17]. For this reason each piece is divided into many sub-pieces (usually 16 KB-sized) which are called blocks or chunks. To reduce the load on seeders (a peer who has access to the whole shared file) a peer downloads pieces not only from the seeders, but also from other peers (which are called leechers).

In this section, we have carefully studied the components of the architecture of BitTorrent based on its specifications [18]. Then we have arranged them into different modules from a developer's perspective. The modules are identified as part of either data plane or control plane, or both. We define the control plane as the part of the architecture which is concerned with drawing up the network map, or handling state oriented messages between other peers or servers. Otherwise, the data plane is defined as the part where the actual data is transferred between the participating peers. Implementation of a full BitTorrent system can be divided into three distinct parts, as follows.