P2P CDNs: A hybrid approach to content distribution

Parth Vidyadhar Natu (pzn5087)
Gaurav Kumar Chandel (gkc5188)

1.0 Introduction

Content Distribution Networks (CDNs), are a network infrastructure built to distribute content provided by content providers to the clients requested on demand while maintaining the downstream latency or quality of service (QoS). The network makes use of multiple cache servers placed at multiple geographical locations. When a user requests the data, the Domain Name Server (DNS) within the requested domain resolves the address to the authoritative DNS of the domain which has the closest cache server placed from the client. The cache server in turn serves that request and since the cache server is picked close to the client location, the QoS is maintained to be high.

CDNs provide the QoS that we aim for, but it comes at a cost. To expand the CDN coverage, we need more number of cache servers which clearly is not very scalable. These distributed cache servers in turn require periodic maintenance which is not cost effective either.

In this report, we surveyed how we can leverage Peer-2-Peer (P2P) architecture combined with CDNs to inherit all the advantages of P2P architecture and not falling for the shortcomings of P2P architecture at the same time. P2P intuitively feels like a natural choice for content distribution networks because it provides scalability by leveraging client resources and cuts down infrastructure cost.

Basically, the expectations from a hybrid P2P CDN are -

- It should be able to scale without inducing additional infrastructure cost.
- QoS should be as good, if not better, as provided by simple CDNs.
- It should not be prone to downsides of P2P architecture.

There are two ways we can integrate CDN and P2P. Either we can build the hybrid architecture over CDN infrastructure or we can go for P2P architecture with CDN servers acting as a backup, the former is called Peer-assisted CDN (PACDN) while the latter is called CDN-assisted P2P (CAP2P). Realizing that building the hybrid architecture over CDN infrastructure is feasible and provides the robustness to our hybrid network, most of the commercially deployed hybrid CDNs are PACDNs and this report will also mostly focus on those.

Clients in a PACDN generally talks to CDN server first which in turn can act as a seeder for the data, a tracker by providing peer list containing list of peers who have this file or content provider by providing the data requested. If the client faces churn in the peers providing content or experiences low bandwidth, CDN server takes over and provide the data without compromising the QoS.

Clients in a CAP2P requests the data directly from the peers (there would be a tracker node to provide with the peer list). If the client's request can't be fulfilled by peers, the request goes to CDN server which serves as a backup and provides the file, maintaining the QoS. As can be observed here, this approach can fall prey to copyright issues, malicious peers etc. and hence is not very popular.

We divided our survey in three categories: **case studies**, **state of the art** of hybrid P2P CDNs and **evaluative studies of P2P-CDN implementations.**

P2P CDNs are already in the market from past couple of years and have been successful in reducing load on servers, achieve larger coverage and reducing maintenance cost [1]. Some of the examples of commercial hybrid CDNs are Peer5, Akamai NetSession, LiveSky, PPLive etc. In this report, we would take a closer look at two of these successful implementations which are Akamai Netsession and LiveSky.

We would also cover what is the current state of hybrid CDNs in the market, emphasizing some common issues that people are skeptical about, like the effect of hybrid CDNs on inter ISP traffic, and workarounds/solutions that most of the hybrid CDN deployments are inhering for them. T will also cover some of the implementation techniques that organization employed to achieve hybrid CDNs.

In the last section, we would present our own implementation of a basic PACDN written using ns3 library which can be used to simulate different scenarios like Inter ISP traffic, application protocols, link capacities and heterogeneity of clients. This was done with the intention of providing a tool which can used to test different hybrid CDN application protocols and compare results. This would enable better application protocol development for hybrid CDNs.

2.0 Case Studies

2.1 Akamai Netsession

Owned by Akamai, Akamai NetSession is one of most successful Peer-assisted CDNs out there [2]. Akamai NetSession has an interface client similar to the bit torrent client. This interface client needs to be downloaded onto the peer machines where it keeps running in the background.
The network is divided into a data plane and a control plane. Servers in the data plane takes part in providing data while servers in the control plane takes part in the maintenance and control logic of the system. Data plane consists of edge servers. The control plane consists of following types of nodes:

- **Connection node:** Maintains TCP connection with the NetSession clients when they are running.
- **Database Node:** Records which peer has which data and the information about the connectivity of the peers.
- **STUN:** These are the nodes that are contacted to find the connectivity of peers which are behind a NAT box.
- **Monitoring node:** These nodes are contacted to record crashes and other sorts of issues reported by peers.

The interface client maintains a TCP connection to the connection nodes while it is running. Any peer requests for data through the interface directs to the control plane which in turn provides a list of peers via the TCP connection which can be contacted to get the data. If the control plane realizes that the request cannot be fulfilled by peers or cannot be fulfilled with expected QoS, it redirects the request to nearest edge server in the data plane which then services it.
The interface client uses HTTP(s) to download content from edge servers. The files are also vetted by edge servers before they can be allowed to pass around. This is done by generating a unique ID and hash for each file by the edge server and sharing this list of IDs and hashes with the clients which they can refer to make sure of the integrity of file. Clients also need to be authenticated by the edge servers over an HTTP(s) connection before they can start participating in P2P transfers.
When a request is made, the DNS redirects the request to the closest CN node to the peer. The CN node in turn refers to the DN in its network to find the peer list for the request. The DN also takes care of checking the connectivity of the peers to exclude peers which are not active, or which are not likely to establish TCP connection with requesting peer because of being behind firewalls or NATs.

Does it degrade network bandwidth of the client? The answer is no. Akamai NetSession employs a back off mechanism which senses the network bandwidth before running. If the network is found contented, the interface backs off and wait for a while to check again. This makes sure that we don’t degrade the network bandwidth of the user.
As per a survey, 54% of the total Akamai traffic is satisfied by the peers.

2.2 LiveSky

The second system that we are going to look at, is LiveSky, a commercially deployed Peer – assisted CDN live streaming system. There have been papers reviewing LiveSky deployment [3], we tried to summarize few of those.
The system contains three main components:

- **Management center**: Responsible for efficient management of LiveSky system
- **Service Nodes or Cache Servers:** Provides content from providers to the client
- **End hosts:** Either LiveSky enabled which can engage in P2P transfers or just usual clients receiving content from service nodes.

The SNs themselves are arranged in a hierarchical structure with lower level of the tier being the closest to the end host. These SNs or edge servers are responsible for providing content to the clients.
The Management Center contains a DNS based load balancing system which works in the similar manner as a tradition CDN query redirection mechanism, it takes into account client location and edge SNs locations and redirects the request to the nearest edge SN.
The service node or the edge server provides multiple functionalities: acting as a traditional cache server for CDN-enabled clients, acting as a tracker to provide a list of peers to LiveSky-enabled client and acting as a seeder for LiveSky-enabled clients.
As we can see, clients in LiveSky system are categorized as either traditional clients or LiveSky enabled clients. There is already an in-built logic to choose which client will act as what. The video stream is divided into multiple segments and a peer is assigned a specific segment to transfer.
LiveySky also takes care of NAT handling. A survey done by PPLive showed that 60%-80% of peers in China are behind NATs. Thus, a mechanism for NAT traversal is required in the system to scale to an even larger population. LiveSky uses STUN for NAT traversal leveraging SNs to act as intermediary to achieve the same.

The other thing that LiveSky took care of was the startup time. Generally P2P based streaming services have the startup time of 30 seconds, LiveSky reduced it to about 15 seconds. It does so by providing initial request segment by the SN itself and when the client buffers get filled, the client is transferred to the P2P layer to receive further segments from peers.
As per a survey, P2P provided for out 42% of the traffic n 2007 during the peak number of active users in China when deployed LiveSky.

3.0 State of the Art

In this section, we would try to summarize all the practical issues [4] faced by the commercial deployments of PACDNs and how have the organizations tried to resolve these -

- Quality of Service: In video streams, the common approach being taken by the operators is to let the CDN servers provide the initial segments before switching over to the p2p overlay to receive further segments.

To take care of the peer churn, CDN servers are used as backups. Example, LiveSky always has one connection maintained with the edge server to fall back to in case the peers leave the swarm in the middle of streaming. In Netsession clients though, two connection are maintained, one with peers and one with the edge server and data is being downloaded from both the connections irrespective of the number of peers joining or leaving the swarm.

- Inaccessibility: As we mentioned earlier, a vast majority of peers are behind firewalls and NAT boxes (90% of clients in China as provided by a survey). Some of these users (as per survey ~30%) are not able to do peer to peer distribution.

For handling issues related to middle boxes, customized version of STUN and UDP protocols are used. These enable the peers behind such middle boxes to communicate with other peers. Though UDP protocol is unreliable, operators such as LiveSky added another layer of ack mechanism in the application layer to enable reliable delivery of data from such peers.

- Cache management: Cache management is one area which is being researched upon aggressively [5]. The client caches can be used to pre fetch popular content such as popular videos etc. for peer to peer distribution. The content can also be cached on peers while streaming which is the case in most video streaming systems. The content can be deleted by the peer or replaced by cache replacement system after a certain period of time.

Example, Kankan, a PACDN, caches 20% of the popular videos on both peers as well as edge servers and rest of the videos only on cache servers.

- Inter ISP traffic: There have been research papers [6] showing the concern over P2P CDNs because of their effect on inter ISP traffic which might not go very well with the service providers.

This problem is generally handled by maintaining locality in the peer list. Systems such Netsession and LiveSky specifically try to choose peers for the peer list which are in the same ISP.

- Authenticity and Copyright Issues: Since the peers can have a file which is illegal and might possess copyright issues, some checks are needed to vet the file before it can be accepted by peers. This is generally achieved by having the CDN servers vet the file before passing it on to the peers. Example, NetSession system have CDN servers generating unique ID and hashes for every file which is shared with every peer to be referred to before accepting any file.

Similarly, clients are authenticated by CDN servers before they can start participating in peer to peer distribution. In LiveSky, there is already a logic in place to decide which client will participate in P2P distribution and which will not.

- Incentives for user participation: This is one of the areas that has not been explored very well. Most of the P2P CDN systems work without any incentive for user participation such as NetSession. While some have deployed a fair bit of incentives for users participating in P2P distributions. An example is LiveSky, which provides high quality video content to the peers who actively and fairly participate in P2P distrubutions.

4.0 Evaluative Studies

We survey [8] to evaluate the working of P2P CDN live streaming solutions. The paper approaches the problem of deploying a P2P CDN solution by using a new approach towards the video buffers used while streaming video (or for that matter any file stream). The paper argues that the need for a Hybrid P2P-CDN architecture is needed to reduce the deployment costs for edge server especially when a popular streamed file is being used by a lot of peers. The handoff to the P2P network greatly reduces the need for a dedicated CDN server, thus reducing cost.
When a video is being streamed, the client maintains a buffer to store the incoming data before it is played. This buffer can be divided based on priority intuitively: It is obvious that the priority of the earlier segment of the buffer is higher than that of a later segment. Thus the buffer is split into two halves : P2P Priority and CDN Priority. Earlier segments of the buffer are assigned to the CDN and later segments are assigned to the P2P network. This ensures CDN-level QoS for video streaming. Figure 2.1 illustrates the format of the buffer.

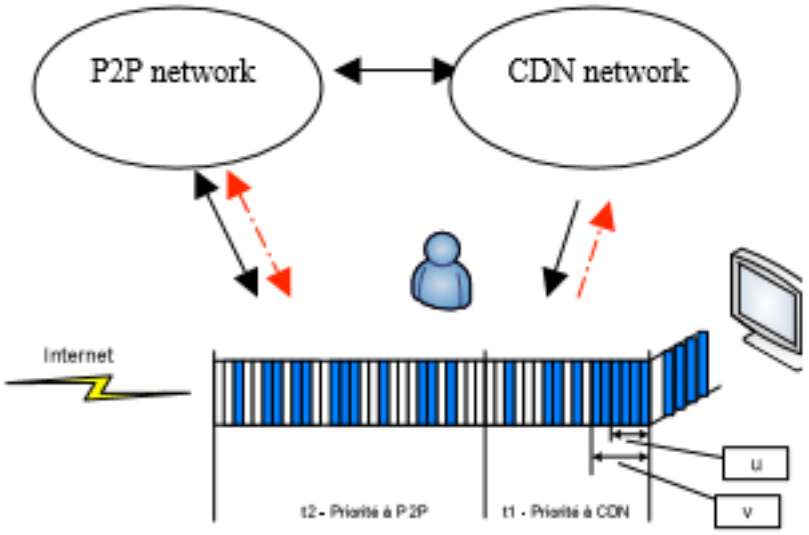

Figure 4.1 : Credits : Reference paper [8]

The paper does not specify a dedicated tracker for the peers and relies on communication of the peers amongst themselves for distributed knowledge of the buffer map.

Working

- Suppose the P2P network is “mature” i.e. the requesting clients for a stream have been registered as peers.
- A requesting client looks at its own buffer map to determine what parts of the video stream have been downloaded, which parts of the stream are supposed to be requested from the CDN and which parts are supposed to be requested from a peer
- The client sends out respective requests for the segments of the stream.
- The client waits for a threshold of contiguous segments (“chunk”) to be able to play the stream.
- Once the threshold is reached, the client plays the stream and the buffer moves forward

Results

[8] uses a simulation workbench called Peersim to simulate the aforementioned algorithm. [8] evaluates the following variables on the network:

- First Delay depending on the number of peers joining the network : The first delay is crucial when a live streaming solution is being deployed as the user needs to play the stream closest to it actually happening live. This delay also persists throughout the stream. When a large number of peers enter the network they use the CDN as a source and this increases the first delay.
- First Delay depending on the percentage of the P2P part in the buffer: The fraction of the buffer assigned to the P2P network determines the "dependence" of the client on the P2P network. The paper observed that in a network with 1 CDN and 1000 peers, the P2P percentage can be increased to 90-95% without packet delay. Furthermore, the first delay is reduced with increase in P2P percentage in the buffer.
- Influence of packet loss on playback of the stream: Playing delay is influenced by packet loss and can be reduced by fallback on the CDN.

In [9] a different approach to P2P CDNs is explored. It involves using a new technology in web browsers: WebRTC. WebRTC stands for Web Real Time Communications and is a new API for the Javascript Engine. It allows two browser sessions to communicate without the presence of a central server to co-ordinate the communication. Two session identify each other using Session Description Protocol (SDP) identifiers. WebRTC also supports communicating over NAT which allows for it to be used for various application scenarios. [9] proposes "WebCDN" -- a novel set of JavaScript libraries to implement a P2P CDN archictecture.
WebCDN libraries consist of two separate JavaScript files – WebCDN server and WebCDN client. These are deployed to the respective machines to construct the CDN network needed to for file transfer.

Architecture

The WebCDN archicture is dividied into two modules: WebCDN server and WebCDN client.

- **WebCDN client:** The WebCDN is responsible for requesting for a resource from the WebCDN server and also to act as a peer when another WebCDN peer asks for a resource. It consists of two separate modules:
  - **Browser Local Storage Module:** This uses the browsers in built key-value storage system to store incoming data. There is no theoretical limit for this as it is only limited by the local storage on the client. This module uses the uint8array data structure in JavaScript as it is more efficient for reads and writes of data.
  - **Data Transfer Module:** This module needs to use parallel asynchronous connections to other peers to avoid latency. Thus, it uses Web Workers in JavaScript to parallely send data requests to other peers. It uses a offer/answer model using the JSEP library. It supports use of DOMString, Blob, ArrayBuffer and ArrayBufferView for data transfers.
- **WebCDN Server:** The WebCDN Server is responsible to find the peers which have the requested resource and to route the connections to these peers. It consists of the following modules:
  - **CometD Management Module:** The CometD library is used for the WebSockets communication for offer/answer requests. As the number of connections for a CometD server is limited, we use a central routing server to store the locations of the peers and their respective CometD server in a global hash table.
  - Data Service Module: This module serves as a mapping between the resources stored and their locations. As this is a key-value storage system, Redis is used as its backend which an efficient C-based key-value DBMS.

Results

The paper measures the following variables while evaluating WebCDN as a technology:

- **Latency:** The setup of connection is a major overhead when operating WebCDN. Thus it is observed that most of the advantage of using WebCDN is achieved by using WebCDN to download large files which eclipse the overhead of the connections setup. WebCDN sets a lower limit of 20KB for transmission.
- **Reduction of Network Traffic:** The WebCDN solution allows it to reduce 48.39% of the traffic while adding and average latency of 147.85 ms per request.

5.0 Implementation

A simplified version of a Peer Assisted CDN was simulated to study the file transfer speeds. The simulation was developed in Network Simulator (ns-3). During the time of research for the project there were no working Peer Assisted CDN network simulations. The “Evaluative Study” papers shown above did not have working simulation code associated with them. Thus, it was decided that the simulation would be built “ground up” using the ns-3 library.

Why ns-3?

ns3 is a general-purpose discrete event network simulator that can be used to simulate any computer network. We used ns3 for the following reasons:

- Easy-to-use APIs
- Ability to obtain both ASCII traces and PCAP traces which can be further analyzed in tcpdump or Wireshark
- Object-oriented abstractions allow for simulation of any network without worrying about low-level details
- Compatibility with most development environments (Linux, Windows, macOS)

Simulation Architecture

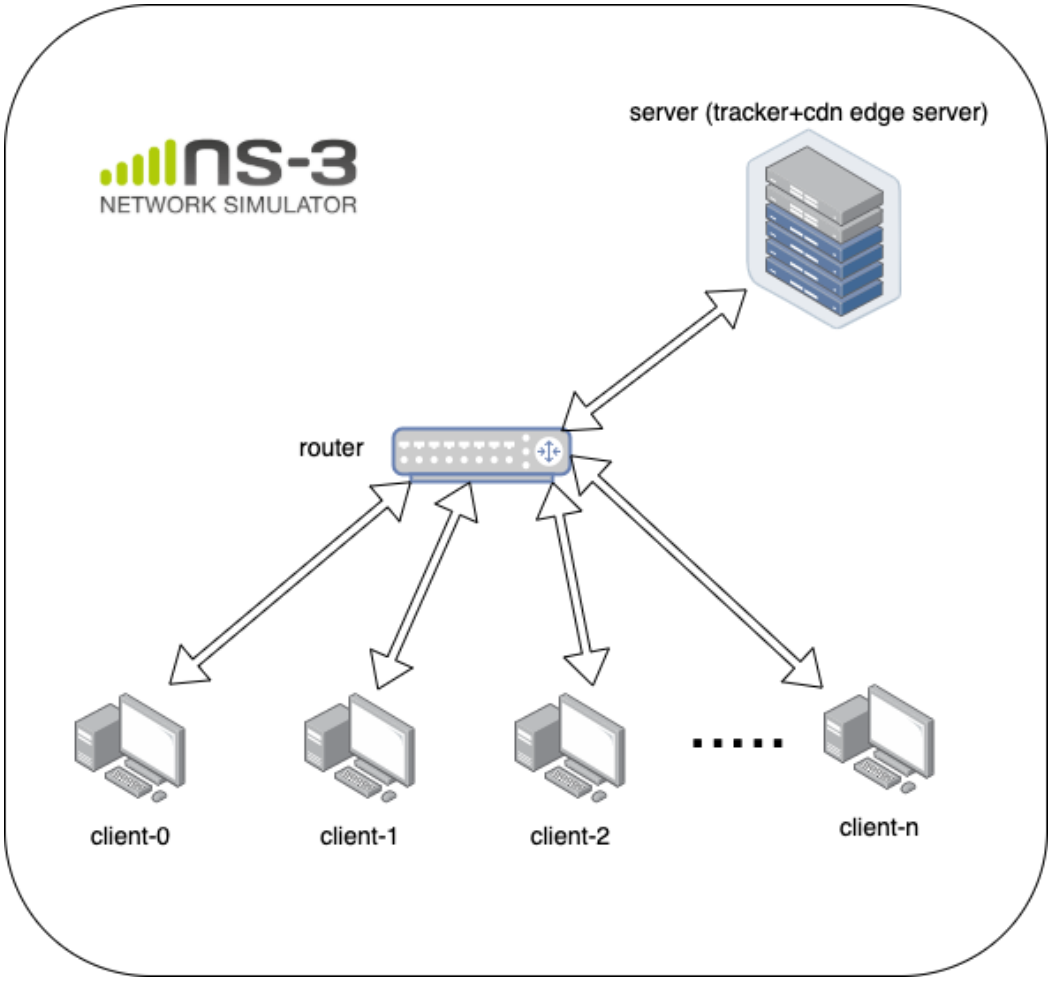

Figure 5.1

Scenario

We implemented a simplified architecture for a Peer-Assisted CDN by abstracting the tracker and CDN in the same node (the “server”). A single router connects both the clients and the server. Each link has a speed of 100Kbps and a propagation delay of 2ms. We use UDP as the transport layer protocol and each link is a PPP link.

Algorithm

- Each packet, except a data packet, is prefixed with a header and flags to identify it at each node.
- **Server-Side:** The server stores a list of nodes that have the data (thus acting as a tracker). It also is the primary source of data when a client requests it.
  - When a client requests a packet:
    - If no other peers are present, the server sends the requested packet and adds the client to the list of peers
    - Else, the server sends a list of peers and waits for an ACK from the client
- **Client-Side:** The client acts as a requesting agent for a packet and as a peer.
  - When a peer requests a packet:
    - The peer will ask for fraction of the entire packet from the client i.e. : *requested_packet_size = total_file_size / number_of_peers*
    - The client sends the requested packet size.
    - Once the peer recieves the packet it will send an ACK packet to the CDN server to update its peer list.

Results

ns3 includes the capability to export the statistics on each link as a PCAP file. We used Wireshark to analyze the PCAP files to obtain the download time for each subsequent client.

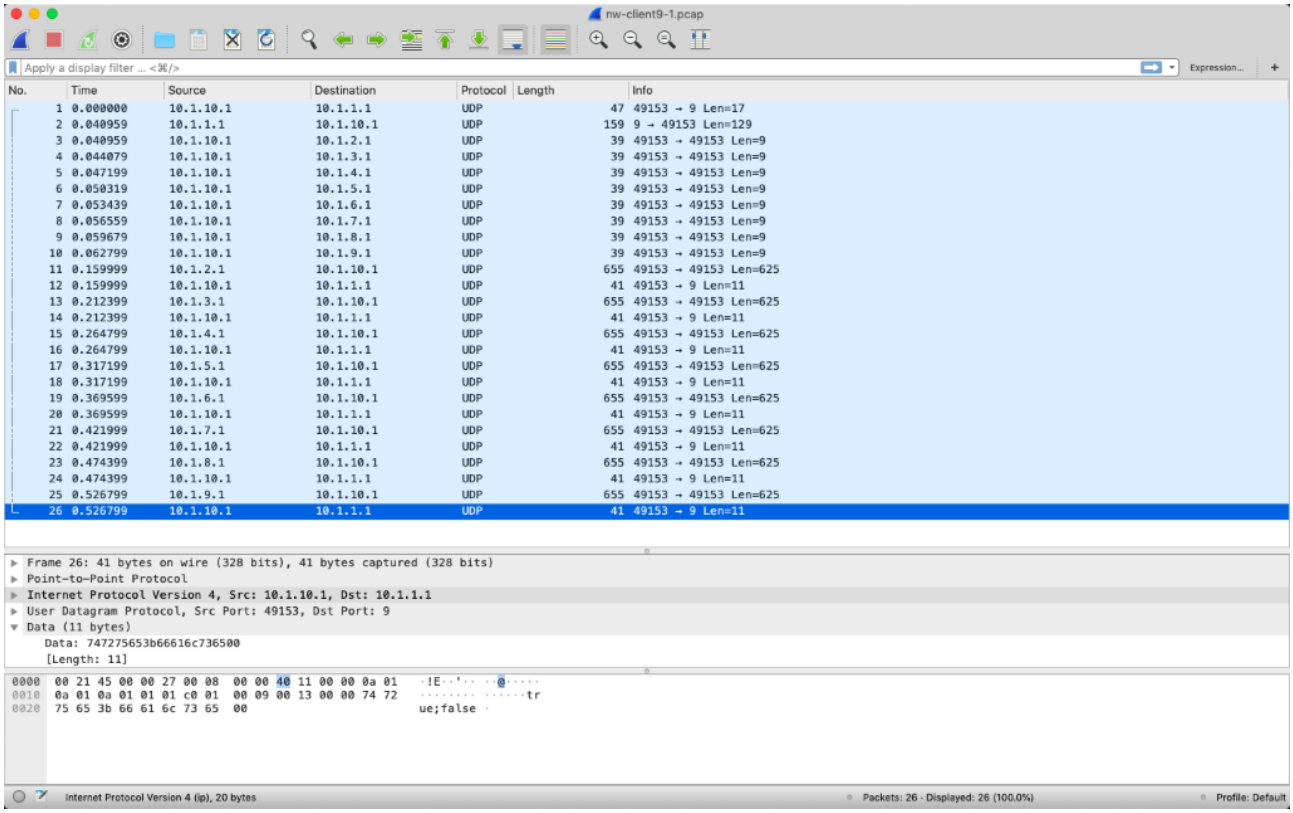

Figure 5.2

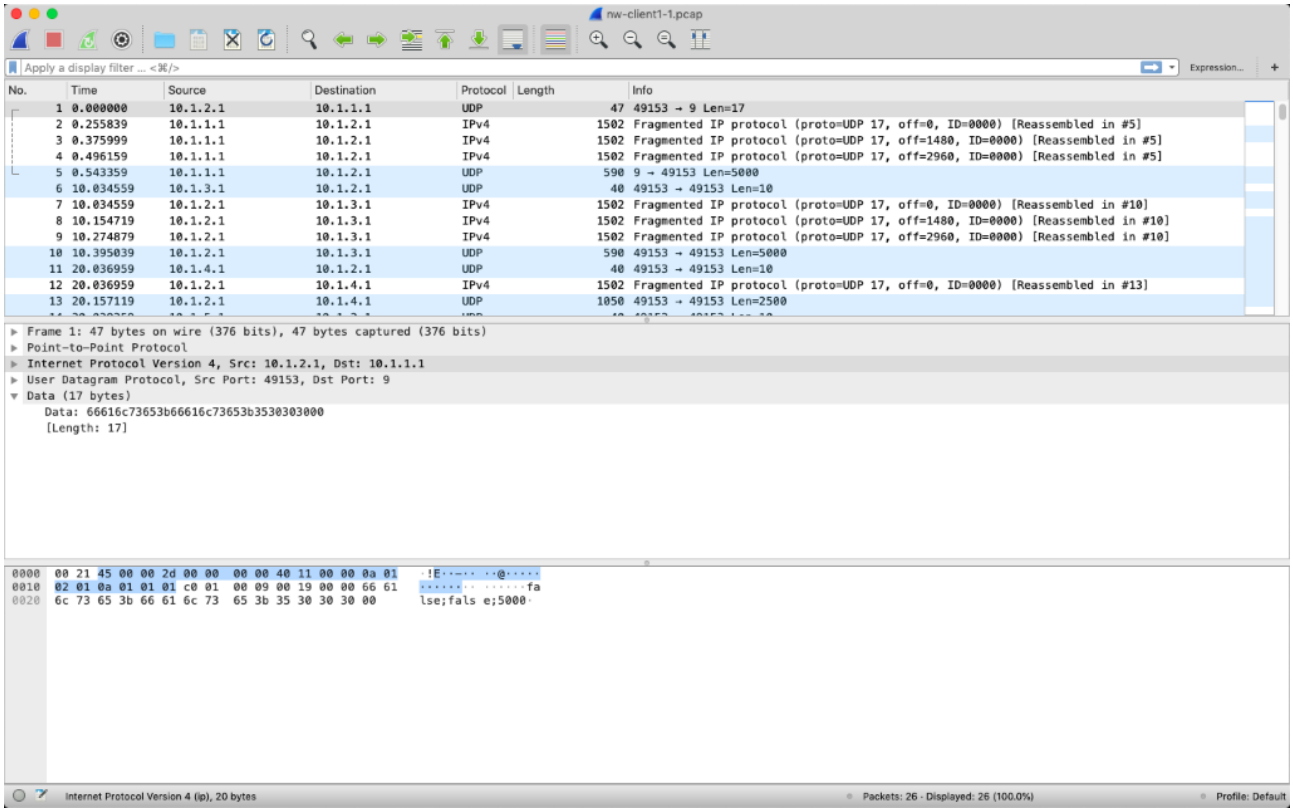

Figure 5.3

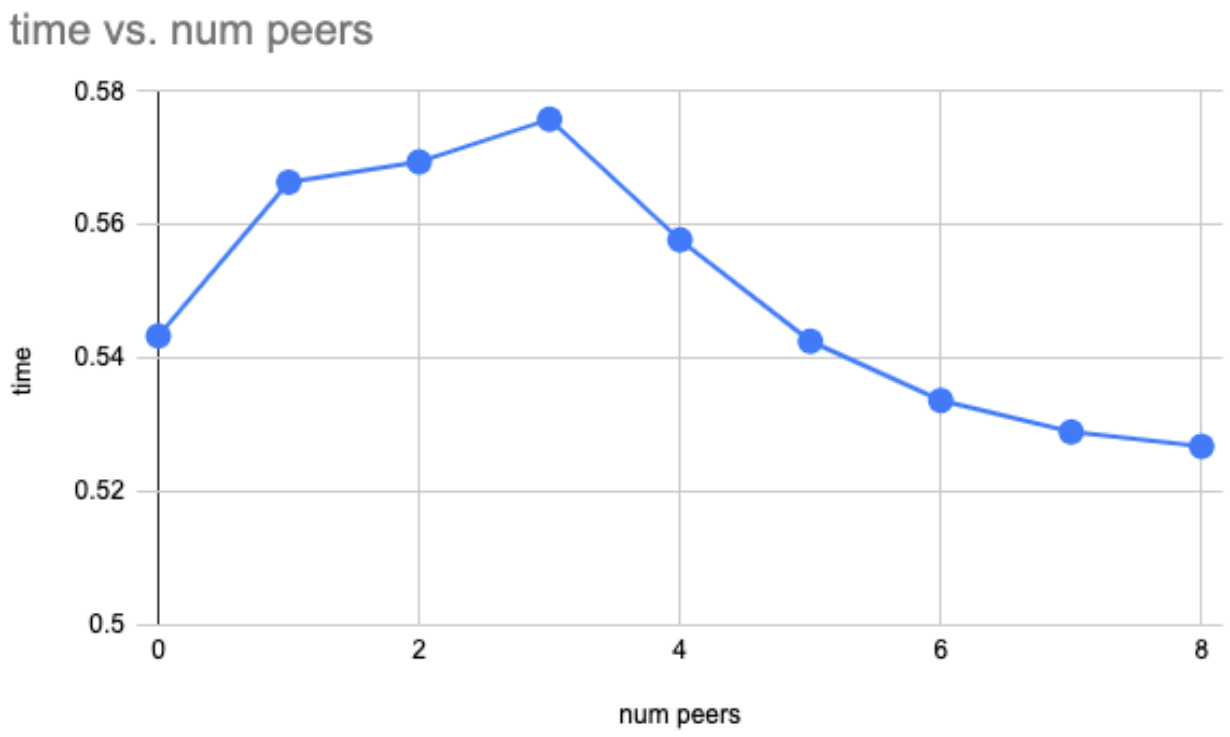

Figure 5.4

Figure 5.2 and Figure 5.3 show the Wireshark PCAP output on two different clients. Figure 5.2 shows a client which as no peers and is requesting the file directly from the CDN. Figure 5.3 shows

a client with 8 peers that host the file. Figure 5.4 shows the download time depending on the number of peers in the network.

As one can see, the download time for a purely CDN approach is very similar compared to the hybrid Peer Assisted CDN approach thus verifying the applicability and practicality of Peer Assisted CDN.

6.0 Conclusion

P2P-CDNs are a viable option towards better usage of network resources as it can efficiently reduce the network load on CDN edge servers by reusing the bandwidth of peers in the network. This also allows commercial deployments to avoid installing more edge servers to serve all the peers as they can communicate amongst themselves, especially for high-demand file.

Peer Assisted CDNs also mitigate various issues that plague traditional P2P networks such as security issues and greedy peers. This allows establishment of commercial networks such as Akamai Netsession and LiveSky.

CDN Assisted P2P networks are an ideal scenario for high efficiency networks but they are theoretically viable without any practical implementation. Their development is still in its infancy.

WebRTC also presents a viable alternative to traditional approaches, especially considering the widespread rate of adoption of modern web technologies.

Thus, Hybrid P2P-CDN network could present the way ahead for modern content distribution networks.

7.0 References

1. https://www.cdnoverview.com/blog/beginners-guide-to-p2p-cdns/

2. *"Peer-Assisted Content Distribution in Akamai NetSession"* - Mingchen Zhao, Paarijaat Aditya, Ang Chen, Yin Lin, Andreas Haeberlen, Peter Druschel, Bruce Maggs, Bill Wishon,Miroslav Ponec

3. *"Design and Deployment of a Hybrid CDN-P2P System for Live Video Streaming: Experiences with LiveSky"* - Hao Yin1 , Xuening Liu1 , Tongyu Zhan1 , Vyas Sekar2 , Feng Qiu3 , Chuang Lin1 , Hui Zhang2 , Bo Li4

4. "*Survey on peer-assisted content delivery networks"* - Nasreen Anjum, Dmytro Karamshuk, Mohammad, Shikh-Bahaei, Nishanth Sastry

5. *"The Role of Caching in Future Communication Systems and Networks"* - Georgios S. Paschos, Senior Member, IEEE, George Iosifidis, Meixia Tao, Senior Member, IEEE, Don Towsley, Fellow, IEEE, Giuseppe Caire, Fellow, IEEE

6. *"Should Internet Service Providers Fear Peer-Assisted Content Distribution?"* - Thomas Karagiannis (U.C. Riverside), Pablo Rodriguez (Microsoft Research), Konstantina Papagiannaki (Intel Research Cambridge)

7. ZhiHui Lu, et al., ―*Scalable and reliable live streaming service through coordinating CDN and P2P* ‖, Proceeding of IEEE 17th International Conference on Parallel and Distributed Systems, IEEE ICPADS2011, pp.581-588.

8. Hoa, Duyen. "*A novel Hybrid CDN-P 2 P mechanism For effective real-time media streaming."* (2009).

9. Shuang, Kai & Cai, Xin & Xu, Peng & Jia, Qiannan. (2015). *WebCDN: A Peer-to-Peer Web Browser CDN Based WebRTC.* 235-243. 10.1007/978-3-319-26979-5_17.

10. Ribeiro, H.B., Lung, L.C., Santin, A.O., Brisola, N.L.: *Web2Peer: implementing a peer-to-peer web browser for publishing and searching web pages on internet*. In: AdvancedInformation Networking and Applications (AINA), Sedona, AZ, pp. 421–428 (2007)

11. Dick, M.El, Pacitti, E., Kemme, B.: *Flower-CDN: a hybrid P2P overlay for efficient query processing in CDN*. In: Extending Database Technology - EDBT, pp. 427–438 (2009)