**ISSN Print:** 2394-7500
**ISSN Online:** 2394-5869
**Impact Factor:** 5.2
IJAR 2019; 5(4): 128-132
www.allresearchjournal.com
Received: 01-02-2019
Accepted: 03-03-2019

**Dr. Dhananjay Dakhane**
Department of Computer Engineering, Ramrao Adik Institute of Technology, University of Mumbai, Mumbai, Maharashtra, India

**Jayesh Kasera**
Department of Computer Engineering, Ramrao Adik Institute of Technology, University of Mumbai, Mumbai, Maharashtra, India

**Pauras Jadhav**
Department of Computer Engineering, Ramrao Adik Institute of Technology, University of Mumbai, Mumbai, Maharashtra, India

**Siddharth Sharma**
Department of Computer Engineering, Ramrao Adik Institute of Technology, University of Mumbai, Mumbai, Maharashtra, India

**Pratik Salunkhe**
Department of Computer Engineering, Ramrao Adik Institute of Technology, University of Mumbai, Mumbai, Maharashtra, India

**Correspondence**
**Dr. Dhananjay Dakhane**
Department of Computer Engineering, Ramrao Adik Institute of Technology, University of Mumbai, Mumbai, Maharashtra, India

# Concurrently streaming media files over P2P DHTs

**Dr. Dhananjay Dakhane, Jayesh Kasera, Pauras Jadhav, Siddharth Sharma and Pratik Salunkhe**

**Abstract**
Nowadays most of people of developing economy countries interact with software every day. As a result of computer systems expansion the problem comes of scalability. Everyday new scalable software are approaching with prob-lems of relia-bility and more up time. Several video streaming services have emerged like Netflix, Hotstar providing concurrent streaming of videos over scalable architecture. This paper presents a method to concurrently stream large media files over a peer-2-peer network using Distributed Hash Tables (DHTs). On demand services are on rise and with addition of every new batch of users, will lead to addition of new infrastructure and servers to serve content. Index TermsP2P, DHT.

**Keywords:** Concurrently, streaming media, P2P DHTs

## 1. Introduction
There are many computer network software systems in the world. Each of them has a different principle of file exchange over the network. P2P based file sharing system is fast to implement in the local scenario. P2P uses host to host based networks. Streaming over P2P enables us to stream fast and concurrently over a layered stable connection. This method has such advantages as Minimal setup and easy to maintain. Low cost of setting up the server. Since each node is master of its own the functioning is independent of the parent node. Peer-to-Peer (P2P) networks are playing an imperative position for providing efficient video transmission over the Internet. Recently, several P2P video transmission systems have been proposed for live video streaming services or video- on-demand services over the Internet. The real challenge lies in streaming the video bit-rate over the P2P connection over the layered architecture. Existing protocols such P2P-TV which refers to peer-to-peer (P2P) software applications designed to redistribute video streams in real time on a P2P network; the distributed video streams are typically TV channels from all over the world but may also come from other sources. The draw to these applications is significant because they have the potential to make any TV channel globally available by any individual feeding the stream into the network where each peer joining to watch the video is a relay to other peer viewers, allowing a scalable distribution among a large audience with no incremental cost for the source. P2P-TV uses client server architecture to serve the clients. Our system proposes a virtual Bit-Torrent client over the web that serves the layered architecture. In a P2P system, each user, while downloading a video stream, is simultaneously also uploading that stream to other users, thus contributing to the overall available bandwidth. The arriving streams are typically a few minutes time-delayed compared to the original sources. The video quality of the stream usually depends on how many users are streaming; the video quality is better if there are more users. The architecture of many P2P networks can be thought of as real-time versions of Bit-Torrent: if a user wishes to view a certain video, the P2P software contacts a tracker server for that channel in order to obtain addresses of peers who distribute that channel; it then contacts these peers to receive the feed. The tracker records the users address, so that it can be given to other users who wish to view the same channel. In effect, this creates an overlay network on top of the regular internet for the distribution of real-time video content.

## 2. Literature Survey
Peer-to-peer (P2P) techniques allow users with limited resources to distribute content to a

potentially large audience by turning passive clients into peers. Peers can self-organize to distribute content to each other, increasing the scalability of the system and decreasing the publishers' costs, compared to a publisher distributing the data himself using a content delivery network (CDN) or his own servers. We dramatically improve peer discovery performance in Bit-Torrent Mainline DHT, the largest distributed hash table (DHT) overlay on the open Internet.

A DHT is a distributed data structure that associates keys with values, just like traditional hash tables or hash maps and is generally built as a self-organizing overlay network of nodes. DHTs are mainly used in large scale applications by providing services to add, delete and look-up hash keys with very good scalability and performance on distributed environments consisting of many thousands of nodes.

Raul Jimenez [1] in his thesis illustrated the scalability issues occuring on P2P networks including that connectivity artifacts are common on the underlying network (i.e., the Internet) used by the Mainline DHT overlay. Furthermore, these connectivity artifacts have the potential to pollute rout- ing tables and degrade lookup performance. Usage of DHTs such as Kademlia has increased performance lookup of the DHTs and nodes associated. Sergio Bessa [2] in his thesis illustrated the development, implementation and simulation of a simple Distributed Hash Table (DHT) protocol for a Peer to peer (P2P) overlay network inspired by small world [3, 2] concepts. Chord DHT was used in the implementation for the P2P network with access and retrieval of data from the peers on the same. Mike Freedman [3] in his course on Com- puter Networks illustrated several algorithms used for P2P networks along with the design goals and the problems faced in system design of each. Issues such as Heterogeneity, False Nodes and Underlay network issues were illustrated.

Jeonghun Noh and Sachin Deshpande [4] proposed a pseudo-DHT, a descent distributed search algorithm to lo- cate contents in a P2P time streaming system. The search algorithm provides a server free content discovery, which allows systems to become more distributed, thus avoiding a single point of failure in the system. In our approach, a distributed lookup overlay is constructed on top of the peers. This lookup overlay provides a foundation for pseudo- DHT services. Pseudo-DHT services include registration, retrieval, and deletion; With registration, peers register their cache information (key) and their network address (value). Registered information is retrieved later for other peers seeking a random video block (retrieval). When peers leave a system, their registration information is discarded (deletion). Pseudo-DHT is a variant of DHT, a distributed version of a hash table. Pseudo-DHT is different from classical DHTs in the following aspects:

- Register (key, value) may perform the alteration of a given key when a key collision occurs.
- Retrieve(key) may return a value associated with a given key or a key closest to the given key, referred to as best effort search.

They suggested a system to apply pseudo-DHT to P2T- SS [5]. P2T-SS is a novel P2P streaming system which can provide live and time-shifted streams. Time-shifting allows viewers to watch a live stream with an arbitrary offset at a later time. It also allows viewers to pause and resume video playback. In P2T-SS, a live video stream is segmented into blocks. Peers store video blocks that correspond to a continuous portion of a video in their local buffer. Peers perform a registration of the first video block in the buffer instead of all video blocks in the buffer. This reduces the overhead of holding keys and makes peers available to other peers earlier.

## 3. Existing Algorithm

### A) Pseudo DHT

P2P streaming systems can be classified into the fol- lowing types: (1) server-dependent type, (2) hybrid type [4], and (3) fully distributed type. In Type (1) systems, peers provide their resources to the system with the construction and maintenance of data delivery overlays controlled by central servers. In Type (2) systems, peers actively participate in constructing and maintaining data delivery overlays. How- ever, there may be a central server that assists peers in some aspects, such as locating peers with contents in their inter- est. In Type (3) systems, peers are fully in charge of the distribution of media data among themselves as well as seeking supplier peers. We are proposing a Type (3) system which uses a Pseudo DHT [1] algorithm. Pseudo-DHT [1], a descent distributed search algorithm to locate contents in a P2P time- streaming system. This search algorithm provides a server- free content discovery, which allows Type (1) or (2) systems to become more distributed, thus avoiding a single point of failure in the system. In our approach, a distributed lookup overlay is constructed on top of the peers. This lookup overlay provides a foundation for pseudo- DHT services. Pseudo-DHT services include registration, retrieval, and deletion; With registration, peers register their cache information (key) and their network address (value). Registered information is retrieved later for other peers seeking a random video block (retrieval). When peers leave a system, their registration information is discarded (deletion). Pseudo-DHT is a variant of DHT, a distributed version of a hash table.

### B) Registration

Peers register the index of the first video block they store in the buffer. Let block index i denote a key. i is hashed using SHA-1, a base hash function. Like the node ID, the hashed value of the key ranges from 0 to 2m1 (and mapped into [0, 1) with normalization). The space of (normalized) keys and node IDs is referred to as a key/node space. After registration.

We choose to use hashed block indexes over block indexes themselves as a key for the following reasons. First, it is difficult to spread out keys over a key/node space with- out knowing the length of a media stream a priority. For in- stance, when a program is broadcast live, its time length is not known beforehand. Second, continuous video blocks are spread out over peers that are not adjacent to each other the video chunks from being permanently lost due to node failure. This enhances resource availability in the system. Once a peer computes a hash value, it executes register (key, value), where value is its network address. When multiple peers attempt to register with the same key, keys may become associated with multiple values. This is called key colli-sion. To prevent a key from being registered with multiple values, successive key modifications are performed by the registering peer until no key collision occurs. Registration with successive key modification. Key is a hashed index of a video block. value represents the network address of the peer associated with the key. Off set is added to a key when a key collision occurs. If the offset is of positive value, a subsequent attempt is made forward in the key/node space. If the offset is negative, the attempt is made backward in the key/node space

When a registration is successful, a peer can start to fill their buffer. If a forward key modification is made, the peer starts to buffer video after the video block with the modified key becomes available. If a backward key modification or no key modification is made, a peer can start to buffer video immediately after it finds a supplier peer. Fig. 1 depicts the two different approaches of successive key modification for registration: forward and backward.

Registration with successive key modification

- N Attempt = 0
- if Register(key, value) fails then
- key = key + offset nAttempt = nAttempt + 1
- if key ¡ 0 then Register With Force (0, value)
- else if n Attempt = Thres then
- Register With Force(key, value)
- end if
- end if

**C) Retrieval**

Peers register the index of the first video block they store in the buffer. Let block index i denote a key. i is hashed using SHA-1, a base hash function. Like the node ID, the hashed value of the key ranges from 0 to 2m1 (and mapped into [0, 1) with normalization). The space of (normalized) keys and node IDs is referred to as a key/node space. After registration.

We choose to use hashed block indexes over block indexes themselves as a key for the following reasons. First, it is difficult to spread out keys over a key/node space with- out knowing the length of a media stream a priority. For in- stance, when a program is broadcast live, its time length is not known beforehand. Second, continuous video blocks are spread out over peers that are not adjacent to each other the video chunks from being permanently lost due to node failure. This enhances resource availability in the system. Once a peer computes a hash value, it executes register (key, value), where value is its network address. When multiple peers attempt to register with the same key, keys may become associated with multiple values. This is called key colli- sion. To prevent a key from being registered with multiple values, successive key modifications are performed by the registering peer until no key collision occurs. Registration with successive key modification. Key is a hashed index of a video block. value represents the network address of the peer associated with the key. Off set is added to a key when a key collision occurs. If the offset is of positive value, a subsequent attempt is made forward in the key/node space. If the offset is negative, the attempt is made backward in the key/node space.

- Retrieval with successive key modification
- nAttempt = 0 if Retrieval(key) fails then
- key = key + offset nAttempt = nAttempt + 1
- if key ¡ 0 or nAttempt = Thres then
- Return NULL
- end if
- end if

When a registration is successful, a peer can start to fill their buffer. If a forward key modification is made, the peer starts to buffer video after the video block with the modified key becomes available. If a backward key modification or no key modification is made, a peer can start to buffer video immediately after it finds a supplier peer. Fig. 1 depicts the two different approaches of successive key modification for registration: forward and backward. Once a supplier with available bandwidth is found, the supplier starts to deliver video from video block.

If no appropriate supplier is found the peer connects to the server to receive the video. Forward successive look up

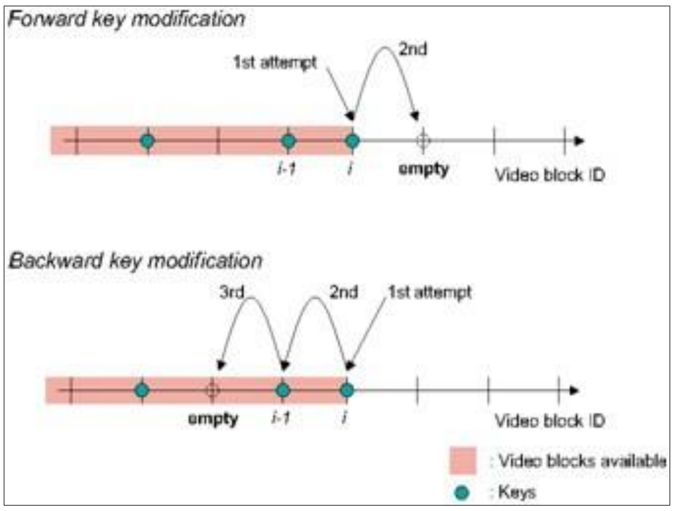

**Fig 1:** Key Modification

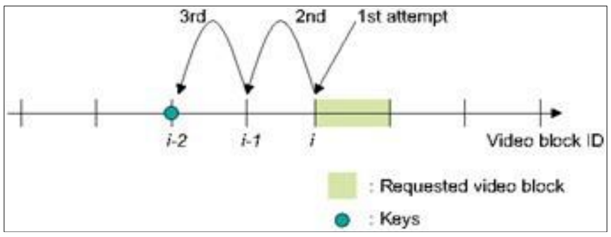

**Fig 2:** Back Key Modification

Is not appropriate because the cache contents of a supplier needs to span the requested video block i. Figure 2 depicts the retrieval with backward successive key modification. As we view the value in (key, value) pair as a pointer to a peer, the lookup overlay is independent of a data delivery overlay. In other words, pseudo-DHT provides a basic functionality to store and seek random video blocks. There are a number of approaches to enhance retrieval performance. One approach is to send simultaneous retrieval requests. In simultaneous retrieval, n retrieval attempts are made in parallel with key i,i+1, ..., and i(n-1), respectively. When more than one reply is received, the earliest reply may be used to reduce the retrieval latency.

## 4. Proposed Methodology

The web based architecture supports multiple DHTs which stores the users file hash. The hash is stored in the DHT which is generated using the SHA-1 over the network. This hash is used to reference the file based in the users system. The secure 128 bit hash contains the offset for the file to be streamed by the other users. Distributed topology such as Chord [2] is used for connecting multiple users over the network. Chord is a scalable, distributed peer-to-peer system that uses a flexible key naming scheme, prepared to support applications as diverse as a Domain Name System (DNS) or

a distributed file storage system. In Chord, participating nodes are disposed in an overlay circle, ordered by the numerical value of SHAl (nodeid), where SHAI is the well-known SHA-1 cryptographic digest. This ordering of the participating nodes induces a natural partitioning of the participating nodes.
Chord is used in our system for serving the following uses:

- This scheme allows to balance the load between many computers instead of a central server to ensure avail-ability of their product.
- Once a computer joins the network its available data is distributed throughout the network for retrieval when that computer disconnects from the network. As well as other computers data is sent to the computer in question for offline retrieval when they are no longer connected to the network. Mainly for nodes without the ability to connect full-time to the network.
- Retrieval of files over the network within a search- able database.
- Retrieval of keys is faster and finding of the successor to the key is efficient.

The key value pair in Chord based system helps in locating of nodes along the network. The value in the key are the hashes of files to be streamed over the network. Multiple buckets of values are stored in the key along the network. Once successful connection between the master and the user is set up the Pseudo DHT algorithm is used to stream the video data along with modification in the streamed data to avoid multiple collision of keys in the network.

### A) Workflow

Video.js is used to build the video player for concurrent streaming. The front end is built upon Vue.js which helps in integrating it with any existing project. The flexibility of the framework helps in loading the HTML blocks with a faster loading time as compared to other frontend frame- works. Along with Vue the connection of peer 2 peer with several clients is handled by hls.js which is a JavaScript library which implements the HTTP live streaming client on any video player content served over the internet. It works by transfusing the MPEG-2 Transport Stream along with AAC/MP3 streams into ISO BMFF (MP4 fragments). The key points considered for streaming are:

- Jumping to specific point of the video.
- Jumping to the end point of the video.
- Jump to a last refreshed point of the video.

The video player takes in data using the Web RTC con-nection. Web RTC can be used for multiple tasks, but real-time peer-to-peer audio and video (i.e., multimedia) com-munications is the primary benefit. In order to communicate with another person (i.e., peer) via a web browser, each persons web browser must agree to begin communication, know how to locate one another, bypass security and firewall protections, and transmit all multimedia communications in real-time. Web RTC reduces the time to access the browser communication in a network protected with firewalls.
Using the secure connection the video data can be streamed from the masters file system to the users video player. Go-Signlr was used to signal the other peers in the network for available peers in the network. Once the new

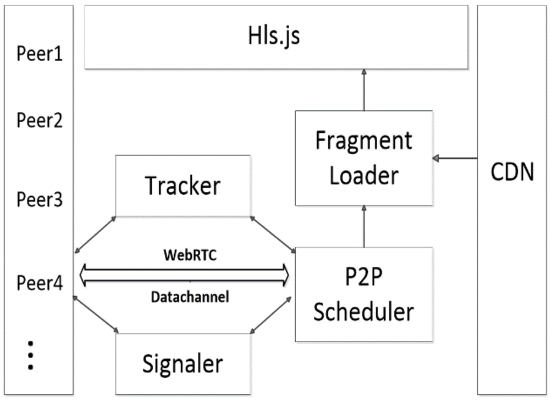

**Fig 3:** Architecture of the System

peer is signaled for the available stream the stream is downloaded from a seeder from the network.

## 5. Results

### A) Parameters

The proposed methodology was implemented on two machines with each having a browser which supports a Web RTC connection. Safari, Chrome & Firefox are available browsers which support Web RTC connection. The media file was uploaded on a central server which servers as main CDN for all the peers. The media file in .mp4 format is converted into a .m3u8 audio file and .ts transport stream file with each frame adjusted to the size of the original video file. The .m3u8 audio file format searches for concurrent streams of .ts files to stream the audio along with the video frames. The system was tested on the following parameters:

- The P2P Ratio which calculated by size of chunks of file uploaded to the channel.
- The Offload which is calculated by dividing P2P down-load size by the total HTTP+P2P downloaded through the channel.

### B) Observations

The above mentioned parameters were tested on a file of size 10MB with over five users and the following points were observed:

- The P2P ratio comes over 100% with user count above five.
- The offload was observed to be half of the size of the video file.
- The first stream of file is served by the HTTP for the new clients with other streams being served from the seeders of the channel.
- A new client joining the network automatically becomes a peer of the channel along with it the old clients become a seeder to this new client.
- The .ts lost due to loss of network was rebuff red by the main server.

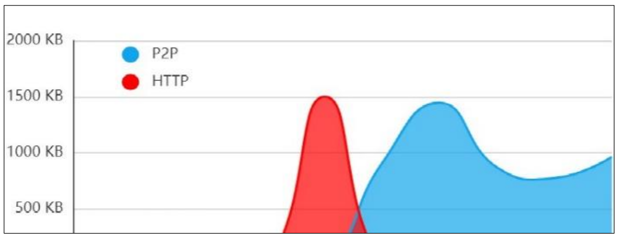

**Fig 4:** P2P vs HTTP Data

## 6. Conclusion

This paper presents use of P2P in video streaming with importance to issues of scalability and reliability on the server based architecture which creates a bottleneck in the current system. Millions of users streaming a file over a network counts to increase of availability of the server. Our system proposed a method to accommodate and increase the reliability of the streaming experience with increase in the users. Video streaming is concern of all major technology firms along with applications such as Video Chat, Video Transfer and Video streaming has created a potential market which can be utilized by using P2P as a service to end user.

## 7. References

1. Raul Jimenez. Distributed Peer Discovery in Large-Scale P2P Streaming Systems, Doctoral Thesis in Communication Systems Stock- holm, Sweden KTH School of Information and Communication Technology, 2013.
2. Sergio Bessa. Storage and retrieval on P2P networks: A DHT based protocol, IEEE, 2007.
3. Mike Freedman. Peer-to-peer systems and Distributed Hash Ta- bles (DHTs), COS 461: Computer Networks Spring 2009 (MW 1:30- 2:50) in COS 105, University of Princeton.
4. Jeonghun Noh and Sachin Deshpande. Pseudo-DHT: Distributed Search Algorithm For P2P Video Streaming, Tenth IEEE International Symposium on Multimedia, 2008.