# A Review paper on Social-Network-Aided Video-on-Demand for peer to peer Network

**Ms.Virangana S. Waghchaure[1], Prof. Mr.Viresh Chapte[2]**
[1, 2] Department of Computer Engineering
[1, 2] Dr.D.Y.Patil School of Engineering & Technology, Lohegaon Pune

***Abstract****- In order to cost-effectively serve a large number of users, video-on-demand (VoD) content providers often place distributed servers close to user pools. These servers have heterogeneous streaming and storage capacities, and collaboratively share contents with each other. A critical challenge is how to optimize movie storage and retrieval so as to minimize system deployment cost due to server streaming, server storage, and network transmission between servers. Using a general and comprehensive cost model, we propose a novel VoD architecture using linear source coding. All the movies are source-encoded once at the repository, by coding k source symbols of movie m to n (m) source-coded symbols. These coded symbols are then distributed to the servers. We optimize n (m) and the number of symbols to retrieve from each server for a request. Our solution approaches asymptotically to global optimum as k increases. We show that even when k is low (say, 30), near optimality can be achieved. Furthermore, the solutions on n (m) , symbol distribution and retrieval can be efficiently computed with a linear program (LP). Through extensive simulation, our algorithm is ;shown to achieve substantially the lowest cost, outperforming traditional and state-of-the-art heuristics by a significantly wide margin.*

***Keywords****-* Peer-to-peer (P2P) P2P networks, social networks, Distributed video-on-demand (VoD) cloud.

## I. INTRODUCTION

Distributed video-on-demand (VoD) has emerged as an important and lucrative cloud service. In order to provide such service in a cost-effective manner scalable to large number of users, a content provider often deploys distributed servers close to user pools. These servers cooperatively store and retrieve movies depending on their popularity. It minimize the total deployment cost by optimizing movie storage and retrieval in the servers.

Fig.1 show Video on demand network with distributed Data server. The network consists of a central Server (repository) storing all the movies and data centers, is also name as proxy servers

Network has central server/repository containing all videos as a whole. Repositories are linked with proxy servers which contains parts of the movies users are directly connected to

Proxy servers. The total concept depends on using the uploading bandwidth of all the users in the network.

**How it works??**

The user will select the video that he wants to download. The user will get a Meta file which will contains the information about the video. Meta file

Contains all the information of the file including its size and the other users that are downloading the video. User will automatically get joined to the network. If there is no server downloading the video then user will get joined directly to the server.

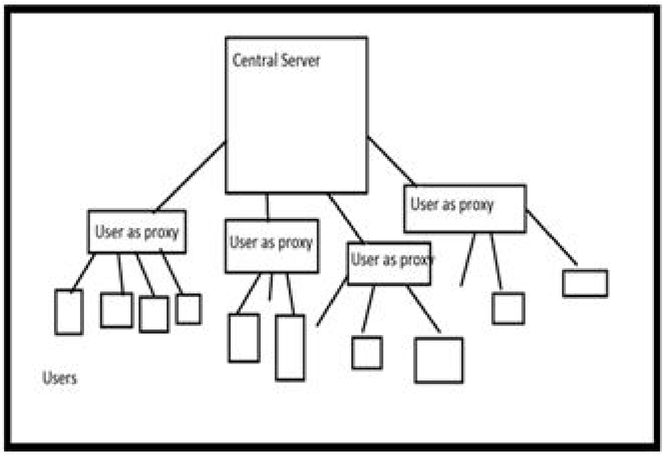

Fig 1. Distributed servers' architecture for Video on demand service.

**How video will get downloaded?**

After downloading the metafile server will divide the video into chunks. Large files are broken into pieces of size between 64 KB and 1 MB. The chunks are decided on basis of size. Now other users that has downloaded the particular chunk can upload to other users. If the chunk is not available with any user then it is directly downloaded to server. If any user uploading content gets disconnected directly from network then the user downloading that part gets directly connected to server in order to avoid time delay.

After download is complete, it will merge all the video parts. And then download will be completed. The transfer speed is affected by a number of variables, including the type of protocol, the amount of traffic on the server and the number of other computers that are downloading the file. If the file is both large and popular, the demands on the server are great, and the download will be slow.The transfer is handled by a protocol (a set of rules), such as FTP (File Transfer Protocol) or HTTP (Hypertext Transfer Protocol). This will solve the problem of leeching.

**Pieces and Sub-Pieces** A piece is broken into sub-pieces ... typically 16KB in size. Until a piece is assembled, only download the sub-pieces of that piece only. This policy lets pieces assemble quickly

**Pipelining** When transferring data over TCP, always have several requests pending at once, to avoid a delay between pieces being sent. At any point in time, some number, typically 5, are requested simultaneously. . Every time a piece or a sub-piece arrives, a new request is sent out.

**Piece Selection** The order in which pieces are selected by different users is critical for good performance .If an inefficient policy is used, then peers may end up in a situation where each has all identical set of easily available pieces, and none of the missing ones.If the original seed is prematurely taken down, then the file is directly connected to server.

Linear source coding is commonly applied for constrained optimization.
The constraint are limited resources.The main elements for source coding are
1. Variables
2. Objective functions
3. Constraints
4. Variable bounds.
In short it is planning with linear models.

**Contributions have three-folds:**

- General and comprehensive consideration of bandwidth and storage for video-on-demand:

Video on demand considers the inter-dependency among server bandwidth, server storage and network traffic in cost optimization.Now video on demanding network capturing all these parameters. Their cost model is hence more general and comprehensive.

- **Bucket-filling:** A novel movie distribution and retrieval algorithm with source coding:

Novel video on-demand network using linear source coding. A requestfor a movie can be satisfied by filling a bucket of size ksymbols. It termed as bucket-filling, is remarkablySimple and effective for movie distribution and retrieval.

- **Asymptotically optimal performance for distributed video on-demand:**All the movies are source-encoded once at the repository, by coding k source symbols of movie m to n (m) source-coded symbols. These coded symbols are then distributed to the servers. They optimize n (m) and the number of symbols to retrieve from each server for a request. The solution approaches asymptotically to global optimum as k increases. We show that even when k is low (say, 30), near optimality can be achieved. Furthermore, the solutions on n (m), symbol distribution and retrieval can be efficiently computed with a linear program (LP). Through extensive simulation & comparison bucket filling algorithm is achieve substantially the lowest cost, outperforming traditional and state-of-the-art heuristics by a significantly wide margin

We study previous work as follows. Many Heuristics had proposed for movie replication and retrieval [1–6]. These algorithms are generally sub-optimal andtheir performance bounds are not easy to analyze or derive. Incontrast, bucket-filling achieves asymptotically optimal performance by increasing the parameter k. For the work studying The cost issue of Video on demand D [1, 4, 7–9], they often have not sufficiently considered the more general case with network access cost, storage constraint and streaming cost of the servers. Our model captures all these elements, leading to a more complete, realistic and practical formulation.

## II. BUCKET-FILLING ALGORITHM

### 2.1. System description

A movie m is source-coded only once at the repository by taking k source symbols to generate $n^{(m)} \geq k$ equal-sized coded ones. As a user has to collect k symbols in order to decode the video, k is a tunable system parameter depending on the level of coding delay and complexity the provider is willing to tolerate. Out of the n(m)coded symbols, the repository stores any of the k coded symbols, and distributes the remainder without replication to the proxy servers. In the network, movies are distributed and retrieved according to the following

**Coded Symbol Distribution:** The repository encodes the movies once and then distributes the coded symbols of the

movies to each server. The symbol distribution needs to be done only upon major system changes, e.g., upon the introduction and removal of movies or change in movie popularity
Which affect movie storage in a major way.

**Coded Symbol Retrieval:** A movie request carries a "bucket" of size k symbols. If its home server has not stored, and hence cannot supply, enough k symbols to serve the request, it "pulls" the missing ones from the other proxies or central servers. Through this bucket-filling mechanism,the servers cooperatively store and supply symbols on-demand with each other to fulfill requests.

We compare bucket-filling with the following traditionaland recent movie replication schemes:

- Random, where each server randomly stores movies withoutconsidering their popularity. This is a simple storagestrategy.
- MPF (Most Popular First), where each server stores themost popular movies. This is a greedy strategy, but doesnot take advantage of cooperative replication.
- Local Greedy [1], which divides the movies into three categories,those popular ones which all servers store (full replication), those medium popular ones which only oneproxy server store (single copy), and those unpopular ones

Which only the repository stores (no copy). By formulating an LP problem, it seeks to minimize network cost. As Local Greedy assumes homogeneous access cost, we set its access cost to be equal to the average access cost between servers in our network.

Form all the comparison schemes, upon a miss request, the home server v chooses an available server u which has the requested content with probability proportional to 1/Cuv. It isa reasonable, simple and effective strategy because the server with lower access cost has higher chance to be chosen. Withthis probabilistic approach, a server with low access cost is not always selected so as to avoid congestion

## III. CONCLUSION

In this work, we have studied optimal movie distribution and retrieval to minimize deployment cost for video-on-demand (VoD) with distributed servers. The deployment cost captures the costs of server streaming, server storage and network transmission cost. We have studied a VoD networkusing source coding which asymptotically achieves exactly optimum depending on a coding parameter. Movies are distributed and retrieved efficiently using "bucket filling" algorithm. . We are study extensive compare the performance of bucketfilling with other traditional and state-of-the-art schemes. And results show that bucket-filling achieves its close optimality with substantially much lower cost, and outperforms the other schemes by a wide margin (multiple times in many cases, and more than 100% in most cases).

## ACKNOWLEDGMENTS

The authors gratefully acknowledge the contributions of [1] Zhangyu Chang and S.-H. Gary Chanfortheir work on Bucket-Filling: An asymptotically optimal video-on-demand network with source codingThank to all for theirguidance, help and timely support .It gives me great pleasure and immense satisfaction to present this paper.

## REFERENCES

[1] Zhangyu Chang and S.-H. Gary Chan, Senior Member, IEEE Bucket-Filling: An Asymptotically Optimal Video-on-Demand Network WithSource CodingIEEE TRANSACTIONS ON MULTIMEDIA, VOL. 17, NO. 5, MAY 2015

[2] S. Borst, V. Gupta, and A.Walid, "Distributed caching algorithms for content distribution networks," in Proceedings IEEE INFOCOM 2010, Mar. 2010, pp. 1–9.

[3] Zhuolin Xu ; Ning Liu"Optimizing video-on-demand with source coding"Chan, S.-H.G. Dept. of Comput. Sci. & Eng., Hong Kong Univ. of Sci. & Technol., Kowloon,China .Multimedia and Expo (ICME), 2013 IEEE International Conference on15-19 July 2013

[4] C. Huang, J. Li, and K. W. Ross, "Can internet video-on-demand be profitable?," in Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun. New York, NY, USA, 2007, pp. 133–144.

[5] S.-H. G. Chan and F. Tobagi, "Distributed servers architecture for networked Video services," IEEE/ACM Trans. Netw., vol. 9, no. 2, pp.125–136, Apr. 2001.

[6] D. Wu, J. He, Y. Zeng, X. Hei, and Y. Wen, "Towards optimal deployment of cloud-assisted video distribution services," IEEE Trans. Circuits Syst. Video Technol., vol. 23, no. 10, pp. 1717–1728, Oct. 2013.

[7] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand

applications,” in Proc. IEEE INFOCOM, Mar. 2012, pp. 460–468.

[8] Y. He, I. Lee, and L. Guan, “Distributed throughput maximization in P2P VoD applications,” IEEE Trans. Multimedia, vol. 11, no. 3, pp. 509–522, Apr. 2009.

[9] S.-H. Gary Chan, “Operation and cost optimization of adistributed servers architecture for on-demand video services,”IEEE Communications Letters, vol. 5, no. 9, pp.384–386, Sept. 2001.

[10] Yung R. Choe, Derek L. Schuff, Jagadeesh M. Dyaberi, and Vijay S. Pai, “Improving VoD server efficiency with bittorrent,” in Proceedings of conference on Multimedia (MULTIMEDIA ’07), New York, NY, USA, 2007, pp. 117–126, ACM.