CHUNK SCHEDULING USING OPTIMIZED MIXED BAYESIAN TECHNIQUE IN P2P VIDEO ON DEMAND

D Sugandhi Mariyal*

R Nithya Devi*

ABSTRACT

Peer-to-peer (P2P) streaming tries to achieve scalability and at the same time it meets real-time playback requirements. The performance of the peer-to-peer streaming systems is the streaming quality they can provide to the peer for the better performance. It depends on the structure of the overlay as well as the way of peers exchange data with each other peers. In this paper, we generate a model based on the Mixed Bayesian Optimization Algorithm that can be used to compare different downloading strategies to random peer selection. We first study two simple strategies: Rarest First (RF) and Greedy. The former technology is a well-known strategy for P2P file sharing that gives good scalability by trying to transmit the chunks of a file to as many peers as quickly as possible. The next strategy is an intuitively reasonable strategy to get urgent chunks first to maximize playback continuity from a peer's local perspective. Next we consider a number of numerical examples on both discrete and continuous problems to illustrate our results and their application to protocol design. Finally we validate our model with simulation and show that our proposed system outperforms the conventional approaches.

Key terms: *p2p, Video on Demand, peer discovery, prefetching.*

* Assistant Professor, CSE Dept, Erode Sengunthar Engineering College,Erode.Tamilnadu.

I. INTRODUCTION

Video streaming over the Internet is already a widely deployed service. The engineering of video streaming from a server to a single client is well studied and understood. This , however , is no Scalable[1] to serve a large number of clients simultaneously. In recent years, a clever solution has emerged, peer-to-peer (P2P) video streaming, which works surprisingly well. The original mechanism is P2P[2] *file sharing*. Each peer obtains an entire file before this possession is known by others. A new kind of P2P algorithm soon got developed, known as P2P file downloading[3]. The most well-known example is Bit Torrent. In this case, the file is divided into a number of *chunks*. In trying to download a file, a peer simultaneously engages in downloading (or, more precisely, sharing) all the chunks of that file.

There are many researches on the engineering of live media streaming[4] from a server to clients along with the wide usage of Internet live streaming in our daily life. The development of encoding and decoding techniques makes many live streaming applications capable of supporting high-quality media streaming. However, serving a large number of users simultaneously challenges the scalability of P2P live media streaming.

In order to improve scalability, a solution based on IP multicast was proposed[5] Multicast routing in datagram Internet works and extended LANs in early 1980's. IP multicast substituted repeatable packets-sending at server by utilizing the routers in the network to manage the distribution and replication of media content from one source to multiple receivers. Popular IP Multicast routing protocols include DVMRP, MOSPF, CBT, PIM-SM etc. Obviously IP Multicast is an efficient solution since all data transmission is performed only once at all links. However, it has many weaknesses. For example, IP multicast is based on IP layer, and it needs the support from network architecture. IP multicast has security and management problems because it is open to any multicasting source. It has difficulties for reliability control and congestion control, because IP multicast provides best-effort service. Application Layer Multicast[6] is regarded as a better solution which was proposed in the early of 1990's. Application layer multicast gains many advantages from its ability that it multicasts data at the application layer by computing, replicating and transmitting at the application-layer end point.

Researchers summarized Application Layer Multicast solutions as three general catalogs: Tree-based, Mesh-based etc. However, no matter which catalog we utilize, Application layer multicast has limited ability to support a large number of users, and it does not solve the scalability problem very well.

Recently, with the rapid development of P2P techniques, live media streaming based on P2P networks becomes a hot topic. As another better solution, P2P technique demonstrates its capacity to solve the scalability problem of live media streaming by supporting a large number of users and high simultaneous demand[7]. Some commercial application are widely accepted by users, for example PPLive and PPStream, and they demonstrate that P2P techniques are another better solution to solve the scalability problem of live media streaming, which is supposed to support a large number of users and high simultaneous demands. The first widely used P2P system is Napster[8] and the early research was mainly focused on the file sharing and distributed hash tables (DHT). DHT is used to construct a special structure for all peers, and hash tables are distributed into each node. Chord, CAN, Pastry and Tapestry belongs to this area. For the P2P file sharing, most applications are widely used nowadays, for example, Gnutella, BitTorrent, KaZaA, eMule. P2P file sharing applications are growing rapidly, which inspires researchers to focus on supporting live media streaming based on the P2P networks. It also inspires researchers to apply similar idea in P2P file sharing that media content is divided into chunks for transmissions based on sharing protocols, for example BitTorrent. P2P networks for live media streaming becomes a hot topic because of its capability to solve the scalability problem of supporting a large number of users.

When P2P techniques for live media streaming were well developed, many scheduling and delivery algorithms were proposed at the same time, for example data-Driven algorithms, mesh-based algorithms, swarming-based algorithms, and pull-based algorithms. These protocols use random process to pick neighbors for a peer when it constructs overlay. For the streaming, they use the similar idea like BitTorrent: media content is divided into many chunks. Each peer periodically sends content information to its neighbors. When peers receive the notifications from their neighbors, they explicitly request chunks from their neighbors who may already store those chunks. In each peer, a buffer window is used to manage the chunk requests, uploading and

downloading. In this way, a peer is a requester who requests media content for playing back, as well as a provider who provides content to its neighbors.

II. RELATED WORK

Peer-to-peer (P2P) streaming tries to achieve scalability meet real-time playback requirements it is a challenging problem. There are two main approaches to this scheduling problem: structured and unstructured[4]. In the first case, the basic idea is to form distribution trees, each a spanning tree from the source to all the peers. The chunks of the file are distributed via different trees in a round-robin fashion. The amount of service each peer provides is related to the total out-degree it has in these spanning trees, and the timing of the service depends on the peer's position in different trees. The challenge of the structured approach is to come up with the distribution trees that fully utilize all the peers, which intuitively will also minimize the delay. All these mechanisms can be implemented as distributed algorithms, as exemplified by BitTorrent[8] and several other systems. Perhaps due to its simplicity (being distributed) and robustness (to peer churn), the unstructured approach is very popular in practice. It is quite unexpected that the seemingly rather chaotic unstructured approach works at all.

III. SYSTEM OVERVIEW

P2P streaming can be thought of as a special case of P2P file downloading. The focus of P2P streaming is no longer only delay and throughput, but also the more stringent playback performance. For this reason, some algorithms that are considered optimal for file downloading may not be optimal for streaming. In the study of P2P content distribution algorithms, whether it is for file downloading or video streaming, practice is leading theory. In practice, chunk-selection, peer-selection, and load-balancing algorithms must all be considered and designed to work together to achieve the best results.

The methodology[9] for evaluation is often based on controlled network experiments, such as PlanetLab, Emulab, or experimental deployment in campus networks. Practical systems are usually designed to be upgradable so that new versions can be tested in real-life environments. In spite of the success of practice, there is still great interest in theoretical models of these P2P

distributed algorithms that are able to provide the insights of why these algorithms work, explain the design tradeoffs, and provide a way to understand the robustness, i.e., the sensitivity of these algorithms to various system parameters.

In the theoretical models of P2P algorithms[10], it is usually not possible to model all the aspects (chunk selection, peer selection, and load balancing) at the same time. To focus on one aspect (or two) only, it is possible to assume an abstract setting in which only one problem is relevant. For example, in studying chunk-selection algorithms, we can assume peer selection is random, and all peers have the same capacity so that there is no need for load balancing. it is assumed that all peers already have all the content so that chunk selection is not needed.

The Zhou–Chiu–Lui model[14], the buffer state of peers; by assuming homogeneous peers, and by making an approximation via an independence assumption, it is possible to write down the probability of buffer occupancy in terms of a set of differential equations. Hence, the continuity, or the playback performance, can be explicitly computed and studied relative to various chunk-selection algorithms and system parameters. This analysis allows us to understand the basic tradeoffs in chunk selection and propose a near-optimal yet practical algorithm.

In this project: 1) to improve the presentation, we reorganize and restate the lemmas and propositions; 2) we discuss the optimality of the proposed algorithms, based on an upper bound; 3) we add a detailed discussion of the contribution of these results by comparing it to some recent and significant relatedworks.

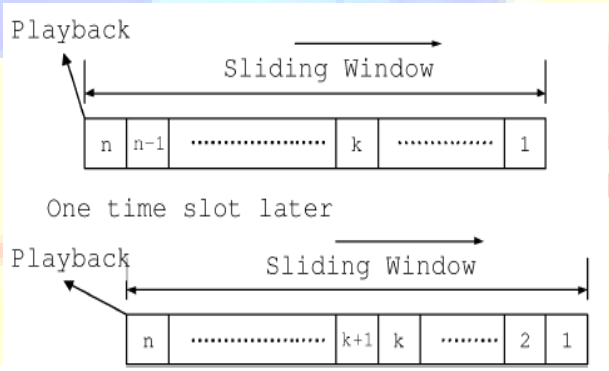

Fig.1. Sliding window mechanism of the buffer B.

Let there be M peers in the network. There is a single server that pushes chunks of (video) content, in playback order, to the M peers. New chunks are generated at the rate of one chunk per

time slot. If the server selects the one peer randomly (to push a chunk) in each time slot, each peer would be receiving new chunks at the rate of 1/M.

Each peer maintains a buffer B that can cache up to chunks[11] received from the network. We refer to the buffer positions according to the age of the chunks stored: B(n) is reserved for the chunk to be played back immediately; B(1) is used to store the newest chunk that the server is distributing in the current time slot. In other words, when the server is distributing chunk (at time t),if t > = n-1 then chunk t – n + 1 is the chunk being played back by that peer. After each time slot, the chunk played back in the previous time slot is removed from B and all other chunks are shifted up by 1. In other words, the buffer acts as a sliding window into the stream of chunks distributed by the server, as shown in Fig. 1. Each buffer space is initially empty and gets filled by the P2P streaming protocol, either from the server or from other peers. The goal is to ensure B(n) is filled in as many time slots as possible, so as to support the continuous video playback.

IV. EXISTING SYSTEM

In the existing systems, they describe a simple stochastic model that can be used to compare different downloading strategies to random peer selection[12]. Based on that model, they study the tradeoffs between supported peer population, buffer size, and playback continuity. To improve playback performance, peers help each other when asked. They modeled the unstructured P2P mechanism as a *pull* process: Each peer selects another peer in each time slot to try to download a chunk not already in its local buffer. They first study two simple strategies: Rarest First (RF) and Greedy.

The former is a well-known strategy for P2P file sharing that gives good scalability by trying to propagate the chunks of a file to as many peers as quickly as possible. The latter is an intuitively reasonable strategy to get urgent chunks first to maximize playback continuity from a peer's local perspective. Yet in reality, both scalability and urgency should be taken care of. With this insight, they proposed a mixed strategy that achieves the best of both worlds. Furthermore, the mixed strategy comes with an adaptive algorithm that can adapt its buffer setting to dynamic peer population.

DRAWBACKS

The main drawbacks of the existing system are that even though their model accomplished for the P2P streaming problem there is an optimal strategy problem is still affecting and they are not focusing on the buffer requirements for P2P streaming. Some existing approaches without the chunk selection had the chance to affect its performance.

V. PROPOSED SYSTEM

Peer-to-Peer video streaming is a scalable approach to serve a large number of clients simultaneously. The original mechanism is P2P file sharing technique. In this case, the file is divided into a number of chunks. The key is that there needs to be a good schedule of which peer is to get which chunk from which other peer at each moment. Thus in our proposed system, a probabilistic model is designed based on the Mixed Bayesian Optimization Algorithm that can be used to compare different downloading strategies to random peer selection. It explores the search space by sampling a probability distribution that is developed during the optimization. Based on this model certain chunk selection strategies are deployed. For each selection strategies, a chunk selection function[13] is described to fill the empty buffer location closest to the playback time. Then we consider a number of numerical examples on both discrete and continuous problems to illustrate our results and their application to protocol design. Finally we validate our model with simulation and show that our proposed system outperforms the conventional approaches.

SYSTEM IMPLEMENTATION

1. Development of the network

Initially we are setting up a model of set of peers for this problem. The difficulty with this approach of setting up a model is how to deal with peer churn and how to get the peers to provide their information reliably for such centralized planning.

2. Chunk Scheduling using Rarest First Strategy

This means the expected rarest chunk is the latest chunk distributed by the server that is missing from all the local peers' buffer[14]. By intention, a peer using the Rarest First strategy will select a chunk that has the least number of copies in the system. Rarest First is very effective

in maximizing peer contribution as the population grows, hence producing good system-wide playback performance. On the other hand, the strength of Greedy is that it takes less buffer space, incrementally, to achieve higher continuity.

3. Chunk Scheduling using Greedy Strategy

We then present the analysis of the Greedy strategy. This strategy aims to fill the empty buffer location closest to the playback time first. the Greedy strategy seems intuitively the best strategy for streaming at the first sight. Through our analysis, we will show that while Greedy[15] may be the best for playback technology from a single peer's point of view, when the peer population is large , it is often too shortsighted from a system's point of view.

4. Chunk Scheduling using Co-operative Strategy

The intuition about the different strengths of the Greedy and Rarest First strategies lead us to propose a Co-operative strategy that can take advantage of both of these chunk-selection algorithms. The basic idea of the Co-operative strategy[16] is to use the front part of the buffer, from position 1 to m, to implement the Rarest First strategy to help distribute the content to many peers as quickly as possible; and to use the tail part of the buffer, from position m + 1 to n, to implement the Greedy strategy to maximize continuity.

For given buffer length and population size, a good question is how to find the optimal m. This can be done by a brute force search since there are only n possible values for m. In practice, there is an adaptive method to search for the suboptimal m in very few steps. This makes it is very easy to implement the Mixed strategy even for dynamic peer populations.

5. Chunk Scheduling using MIXED BAYESIAN OPTIMIZATION

In MBOA, a Bayesian network with local structures in the form of decision trees captures the mutual dependencies among the parent individuals. The first EDA employing the Bayesian network model with decision trees was the hierarchical Bayesian Optimization Algorithm (hBOA)[17]. MBOA is an extension of hBOA from binary to continuous domains. In fact, MBOA is able to deal with discrete and continuous parameters simultaneously, but in this paper we focus on continuous parameters only.

In every generation, the parent population X_{parent} of size $N_{parent} = \tau$. N_{base} is selected from the base population using tournament selection. Then the probability distribution of X_{parent} is estimated and N_{offspr} offspring are sampled. The offspring population is used to replace part of

the base population. For effective diversity preservation, restricted tournament replacement is used.

The probabilistic model M = (T,θ) of the parent population X_{parent} is rebuild in every generation. T = {T_1, …, T_n} is a set of decision trees defining the structural part of the model whereas θ are the quantitative parameters of the model. Each decision tree T_i defines the conditional distribution P ($X_i|\Omega_i$) of the variableX_i, i = 1, …, n. Domain Ω_i denotes the subspace spanned by the variables that affect the value of X_i. The subspace is chosen with regard to all previously generated trees, such that no bidirectional dependencies occur.

VI. PERFORMANCE EVALUATION

The project is implemented with four different algorithms. The algorithms can be implemented in following ways.

Chunk Scheduling using Rarest First Strategy

The Rarest First strategy is the opposite of the Greedy strategy. We know p(i) is an increasing function in i[9]. This means the expected rarest chunk is the *latest* chunk distributed by the server that is missing from all the local peers' buffer. Therefore, the chunk-selection function s(i) for the Rarest First strategy can be expressed as

$$s(i) = \left(1 - \frac{1}{M}\right) \prod_{j=1}^{j=i-1} \left(p(j) + (1 - p(j))^2\right). \quad (1)$$

The meaning of each term is similar as before. The main point is that the search for missing chunks starts from the *latest chunk* B(1), then to B(2), and so on. Again, (1) has a simple form.

Chunk Scheduling using Greedy Strategy

We then present the analysis of the Greedy strategy. This strategy aims to fill the empty buffer location closest to the playback time first. The chunk-selection function s(i), which is the probability of selecting B(i), can be expressed as follows:

$$s(i) = \left(1 - \frac{1}{M}\right) \prod_{j=i+1}^{j=n-1} \left(p(j) + (1 - p(j))^2\right). \quad (2)$$

Since the event that downloading does not occur for a buffer at position is , the probability of this event is hence

$$\Pr[\neg(W(k,j)H(h,j))] = p_k(j) + (1 - p_k(j))(1 - p_h(j)). \quad (3)$$

Equation (3) is based on the event that the server selects other peers to upload, and the chunk selection does not occur for all those positions closer to the deadline than B(i), with the buffer position independence assumption stated earlier. Note, the first term of the (3) equation is the probability the local peer already has the chunk for B(j). The second term is the probability that the local peer does not have the chunk for B(j) and the selected peer (h) does not have that chunk either. The rather complicated formula (2) for s(i) has a surprisingly simple alternative form.

Chunk Scheduling using Co-operative Strategy

The intuition about the different strategies of the Greedy and Rarest First strategies lead us to propose a Mixed strategy that can take advantage of both of these chunk-selection algorithms[20].

Let the buffer B be partitioned by a point of demarcation m, $1 \leq m \leq n$. The Rarest First strategy is used first with buffer spaces B(1), ….., B(m). If no chunk can be downloaded using the Rarest First strategy, then the Greedy strategy is used with the other partition of the buffer, B(m+1), B(m+2), …., B(n). When m = n-1, the Mixed strategy is the same as the Rarest First strategy; when m = 1, the Mixed becomes the same as the Greedy strategy. Through this variation of m, a peer can adjust the download probability assigned for each partition.

The buffer state probability for B(1) to B(m) satisfies the following equations:

$$p(1) = 1/M$$

$$p(i + 1) = p(i) + p(i)\,(1-p(i))^2 \quad \text{for } i = 1, \ldots, m - 1.$$

The probability for B (m+1) to B (n) can be derived by substituting p (1) with p (m) for $i \geq$ m.These equations can be solved numerically.

Chunk Scheduling using MIXED BAYESIAN OPTIMIZATION

To define split nodes in the decision tree T_i, a variable and a split boundary are chosen using Bayesian-Dirichlet metrics[18]. The split nodes hierarchically decompose Ω_i, the domain of P (X_i| Ω_i), into rectangular axis-parallel partitions, Ω_{ij}, j = 1,2,…, that correspond to the leaves of the decision tree. In each leaf, ì -is approximated by a univariate probability density function

using Gaussian kernels. Let $\Omega_{ij} \in \Omega_i$ denote the partition that traverses to the j-th leaf of T_i. Consider all parent individuals that traverse[19] to Ω_{ij}, and let the set $\{x_i\}_j$ denote their realizations of variable X_i. Then the Gaussian kernel distribution in the j-th leaf can be expressed as:

$$P(X_i|\boldsymbol{\omega}_i \in \boldsymbol{\Omega}_{ij}) = \frac{1}{|\{x_i\}_j|} \sum_{m \in \{x_i\}_j} \mathcal{N}(m, \sigma_{ij}^2)$$

All the kernels in the same leaf have the same height $1/|\{x_i\}_j|$ and the same width σ_{ij}. In our experiments

$$\sigma_{ij} = \frac{\max\{x_i\}_j - \min\{x_i\}_j}{|\{x_i\}_j| - 1}.$$

The offspring population X_{offspr} is sampled from the estimated model

$$x_k^{(g+1)} = z_k, \quad z_k \sim \prod_i P(X_i|\Omega_i), \quad k = 1, \ldots, N_{offspr}.$$

N_{offspr} is set to half of the base population size N_{base}, and the fraction of X_{base} selected as parent population is set to $\tau = 0.5$.

Fig. 2. Continuous playback video streaming.

Fig 2 shows the continuous playback video of Optimized prefetching technique which is using the Bayesian Algorithm.

From the Fig. 3. It shows the various prefetching techniques in peer to peer video streaming. In this Greedy technique achieves only 0.3 playback continuity. Using rarest first strategy it achieves 0.5 playback continuity.

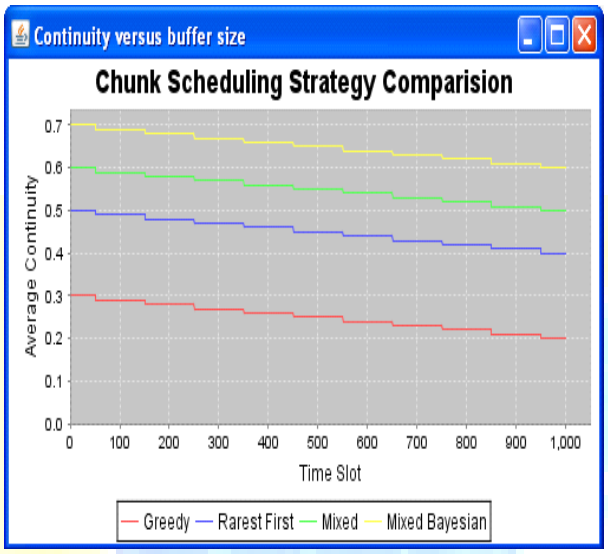

Fig.3. Comparison graph of Various chunk scheduling techniques in P2P Video Streaming.

The proposed Mixed Bayesian technology it achieves 0.7 average continuity in video playback.

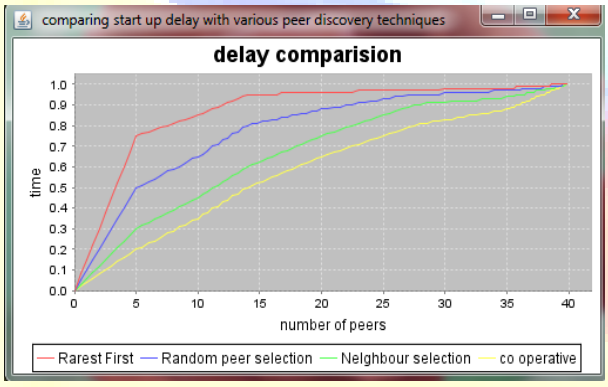

Fig.4. Delay comparison graph with existing techniques.

Fig.4 shows the delay comparison graph between the proposed technique with the existing techniques.

In the existing techniques for the 20 peers they having the more than 8 sec delay, but in proposed technique it have only 5 sec delay. From this the proposed system reduce the delay in peer discovery.

CONCLUSION

In the proposed, Mixed Bayesian Optimization model for the peer to peer chunk scheduling strategy has considered the rarest first, greedy and mixed strategy for the chunk scheduling approach. Later the method of mixed Bayesian optimization for this problem in peer to peer is considered. The performance of the proposed approach with the existing approaches is evaluated. The experimental results also show that the proposed approach works better than the previous methods in case of buffer size, population, and continuity measures.

FUTURE ENHANCEMENT

For the future enhancement work we can also work on some other optimal scheduling strategies which may improvise our proposed approach. Also we can have a further study on reducing the buffering methods in this peer to peer streaming approach. As we have developed an efficient mixed Bayesian optimization problem for the chunk scheduling strategies in the peer-to-peer network, we have to do further studies to make the approach more effective via any other approaches or techniques. We can also approach with peer-to-peer multiplexing based on our proposed method. This method of chunk scheduling can also be presented in the mesh networks for better performance of scheduling with optimal strategies.

REFERENCES

[1] Mol .J.J.D, Pouwelse. J.A, Meulpolder.M, Epema.D.H.J, and Sips.H.J "Give to Get :Free riding resilient VoD in P2P Systems" Multimedia Computing and Networking 2008, Vol. 6818, 681804.

[2] Xuanjia Qiu, Wei Huang, Chuan Wu," InstantLeap: An Architecture for Fast Neighbor Discovery in Large-Scale P2P VoD Streaming",in Department of Computer Science, Sun Yat-Sen University,China,linxl@mail.sysu.edu.cn.

[3] Danqi Wang and Chai Kiat Yeo, "Superchunk-Based Efficient Search in P2P-VoD System",in IEEE transactions on multimedia, vol. 13,no,2, april 2011.

[4] Xuanjia Qiu, Chuan Wu, Xiaola Lin, Francis C.M.Lau."InstantLeap:Fast Neighbor Discovery in P2P VoD Streaming" NOSSDAV'09, June3–5, 2009, Williamsburg, Virginia,USA.

[5] Daqi Wang, Chai Kiat Yeo, "Superchunk based Fast Search in P2P Vod System" Centre for Multimedia and Network technology IEEE Communications Society, IEEE "GLOBECOM" .

[6] Thangamani .R., Siva Kumar. G, "Survey on various peer discovery techniques in P2P VoD system" Radix International Journal for science 2012, Vol:12

[7] Sugandhi Mariyal. D , Siva Kumar. G, "survey On prefetching techniques in P2P video on demand systems" International Journal of Management , IT & Engineering (IJMIE) vol-3, issue-3 , march 2013.

[8] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, And Atul Singh. Splitstream: High-Bandwidth Multicast In Cooperative Environments. In Proceedings Of Symposium On Operating Systems Principles (Sosp) 2003

[9] Yifeng He and Ling Guan " Prefetching optimization in P2P VoD applications" In Proc of International Conference on Advances in Multimedia 2009.

[10] Ubaid Abbasi and Toufik Ahamed " COOCHING: Cooperative Prefetching Strategy for P2P Video on- Demand System".

[11] Osama Abboud,Kontantin Pussep, Markus Mueller,Aleksandra Kovacevic and Ralf Steinmetz "Advanced prefetching and upload strategies for P2P Video-on-Demand" in 2010.

[12] Y.Shen, Z.Liu, S.Panwar, K.W.Ross, and Y.Wang, "On the Design of Prefetching Strategies in a Peer- Driven Video-on- Demand System,:Proc. Of IEEE ICME,pp.817-820, Jul.2006.

[13] J.Mol,J.A.Pouwelse, M.Meulpolder, D.Epema, and H.Sips. GIVE TO-GET: an algorithm for P2P VoD. In MMCN,2008.

[14] B.Zhao, J.Lui, and D.Chiu. Exploring the optimal chunk selection policy for data-driven P2P streaming systems. In International conference on P2P Computing 2009.

[15] C.Zheng,G.Shen,S.Li, "Distributed prefetching scheme for Random seek support in P2P streaming applications". In Proc of ACM workshop on advances in P2P multimedia streaming 2005.

[16] R.Cheng and L.E.K.Bring order to online social networks. In proc of IEEE INFOCOM, 2011.

[17] X.Yang, G.Y., and L.Y. Bayesian-inference based recommendation in online socisl networks. In Proc of IEEE INFOCOM, 2011.

[18] Zhi Wang, Lifeng sun,Shiqiang yang and Wenwu zhu" Prefetching strategy in peer-assisted social video streaming" in 2011.

[19] T.Xu,J.Chen,W.Li,S.Lu,Guo and M.Hamdi,"Supporting VCR like operations in derivative Tree- Based P2P streaming systems",In proc of IEEE ICC'09.Germany ,jun 2009.

[20] Tianyin Xu,Weiwei Wang,Baoliu Ye," Prediction based prefetching to support VCR like operations in Gossip based P2P VoD systems", International conference on Parallel and Distributed Systems In 2009.