Statistical Modeling and Analysis of P2P Replication to Support VoD Service

Yipeng Zhou, Tom Z. J. Fu and Dah Ming Chiu
Department of Information Engineering, the Chinese University of Hong Kong
Email: {ypzhou6, zjfu6, dmchiu@ie.cuhk.edu.hk}

***Abstract*—Traditional Video-on-Demand (VoD) systems reply purely on servers to stream video content to clients, which does not scale. In recent years, Peer-to-peer assisted VoD (P2P VoD) has proven to be practical and effective [1]. In P2P VoD, each peer contributes some storage to store videos (or segments of videos) to help the video server. Assuming peers have sufficient bandwidth for the given video playback rate, a fundamental question is what is the relationship between the storage capacity (at each peer), the number of videos, the number of peers and the resultant off-loading of video server bandwidth. In this paper, we use a simple statistical model to derive this relationship. We propose and analyze a generic replication algorithm RLB which balances the service to all movies, for both deterministic and random demand models, and both homogeneous and heterogeneous peers (in upload bandwidth). We use simulation to validate our results, for sensitivity analysis and for comparisons with other popular replication algorithms. This study leads to several fundamental insights for design P2P VoD systems in practice.**

I. Introduction

Traditional video-on-demand (VoD) is based on the client-server approach. It is expensive and not scalable. In recent years, the Peer-to-Peer approach was first demonstrated to work for live content streaming [2], and later for VoD streaming as well [1]. Various efforts are working on building a P2P-based VoD platform, for example using set-top boxes [3].

VoD streaming is definitely harder to accomplish (than live content streaming) since peers are less likely to have the same content to share with each other. The lack of synchrony (in VoD) is compensated by the following two remedial features in P2P VoD systems: (a) each peer contributes more storage to replicate content; (b) each peer is capable of uploading content different than what is currently consumed (downloaded) locally. The effectiveness of this remedy depends on whether the *right* mix of content is placed at different peers, which is the P2P replication problem at hand. As discussed in [1], P2P replication is a central design issue in P2P VoD systems.

There were various studies of P2P replication in the past, focusing on different goals. For example, some tried to minimize network transmission overhead in replication [4], [5]; others tried to maximize content availability when peers are infrequently on-line [6].

Our problem formulation is based on the requirement on more contemporary P2P VoD systems, where the majority of the peers expect to be streaming the content for immediate viewing. In such systems, video servers must still be deployed to ensure service quality. The P2P network can be viewed as the mechanism used to off-load the video servers. The peers are assumed to be under the control of the content provider (e.g. set-top boxes). The objective is to minimize the server bandwidth. The streaming requirement means there needs to be a balance between the total supply of uplink bandwidth (that is the sum of server(s) and peers' uplink bandwidth) and the total demand (that is the number of viewing peers multiplied by the video playback rate). In practice, the operating regime of particular interest is when the total peer uplink bandwidth is comparable to the demand (of viewing bandwidth). In this regime, ideally the server bandwidth can be zero, if the viewing demand is deterministically known, and all the peers are replicated with the *right* mix of content so as to make full use of their upload capacities. In reality, the impredicability of user demand, and hence the imperfection in content replication and service load balancing will always result in some server load.

The important question is, how does this inevitable server load scale with the critical system parameters, such as the number of peers in the system, the number of movies, and the storage provided by peers? Furthermore, what role can the replication algorithm play? In this paper, we describe a generic model to help settle these questions.

In our basic model, user demand follows a stationary popularity model for movies, and the server is assumed to be off-loaded by a processor-sharing style peer service scheduling. Under this model, we show that the most effective replication algorithm is one that can produce the best load balance among peers, not only in terms of average load, but also in terms of correlation of load. We use a generic algorithm called *Random with Load Balancing* (RLB) to characterize a spectrum of algorithms that can balance the average load, but succeed in different degrees in reducing the correlation of load. This allows us to derive (both a closed-form lower bound and upper bound of) the server bandwidth utilization as a function of the key system parameters. This result not only gives us the scaling properties of what P2P replication can achieve, it also serves as a rule-of-thumb for needed peer storage for given number of movies. Quite amazingly, the result is indifferent to whether peers are homogeneous or heterogeneous (in bandwidth).

Simulation is used to validate our results, study its sensitivity to parameter values and assumptions, and compare RLB with some other intuitive replication algorithms (Proportional and Random). We also study how to turn RLB into a distributed and adaptive algorithm (ARLB), and how to converge to a balanced state given movie popularity. We show the resultant algorithm performs much better than some well-known heuristics for adaptive load, such as the FIFO and LRF (least frequently requested) algorithms.

In the rest of the paper, we first describe our model in section II, the RLB algorithm in section III, and derive our analytical result in section IV. We then propose an adaptive distributed replication algorithm suitable for implementation in section V, and show our simulation results in section VI. Before we conclude the paper, we also discuss related works and explain the significance of our work in light of the earlier publications.

II. Model

We introduce the following notations to describe our model. We use the terms *video*, *movie* and *content file* interchangeably.

- N number of peers in the system
- K number of movies
- η_j relative popularity of movie j, where $\sum_{j=1}^{K} \eta_j = 1$
- U_i upload capacity of peer i, where $\sum_{i=1}^{N} U_i = N$
- L number of movies stored at each peer, where $K \gg L$
- S_j the peer set replicating movie j
- Q_i the movie set stored on peer i
- λ_i the average number of requests received by peer i
- B the average server bandwidth utilization
- M the average number of movie copies, $M = \frac{NL}{K}$.

In the definition of notations, we have assumed all movies are of the same size, and have the same playback rate equal to 1 (normalized to be the same as the average upload capacity of a peer). The peer population is assumed to be fixed (N) and $N \gg K$.[1] Each peer generates a *demand* on the system by downloading movies, and provides a *service* by uploading content to other peers.

When a particular peer selects movie j to download, this demand is assumed to result in a request at all peers in S_j. At any moment, a particular peer i may be entertaining requests from multiple other peers, and it is assumed that i divides its uplink bandwidth equally in serving all outstanding requests. We call this service model *Perfect Fair-Sharing Model* (PFS). Note, this service model implies a fluid assumption, since each peer's uplink bandwidth, no matter how small, is assumed to be divisible. This assumption is crucial in making the problem analytically tractable.

The demand, how a peer selects a movie to download, can be one of the following:

- Deterministic Demand: A fraction η_j of the peer population always accesses movie j and $\sum \eta_j = 1$.
- Stationary (random): The user behavior is described by a transition matrix. In the matrix, the element q_{ij} is the probability that a peer jumps from movie i to movie j after the peer finishes watching the current movie (i). Each peer makes independent decision to select movies. The stationary state can be derived based on the transition matrix. The probability of any peer watching movie j is a Binomial distribution with parameter η_j. Thus, the number of requests for a particular peer is a Binomial distribution, asymptotically (for large peer population) it becomes a Poisson distribution with average number requests $\lambda_i = N \sum_{j \in Q_i} \eta_j$. This is essentially the same user behavior model proposed in [7].

The deterministic demand model is too simplistic. Although in practice the demand model must be dynamic, stationary random demand is a reasonable assumption when demand patterns do not change very fast. The important analytical results in this paper are derived using the stationary demand model.

The objective of the P2P VoD system is to satisfy all peers' playback requirements. Those peers that cannot obtain enough bandwidth from other peers will go to the server as a last resort. The total average server bandwidth utilization B is what we try to minimize. This can be loosely written down as an optimization problem: $\min_{Q_i, \forall i} B,$ subject to $|Q(i)| \leq L, \forall i.$

Note, $(Q_i, \forall i)$ implies $(S_j, \forall j)$, and either one defines the replication allocation, the result produced by a replication algorithm. Due to the definition of the service model, the service capacity of peers can never be exceeded but only *wasted*. Wastage occurs when the total service from other peers received by a peer exceeds its playback rate (which is 1). Because of such wastage, and the fact that we are operating in the regime that the total peer service capacity is barely enough to support the total playback demand, some movies will not receive sufficient service from peers and must rely on the server. So the primary job of the replication algorithm is to find a replication allocation that balance the service bandwidth for different movies, so wastage is minimized.

III. Replication Algorithm

If the demand is deterministic, the replication algorithm's job is straightforward. For the deterministic demand defined in the last section, there is always $N\eta_j$ peers watching movie j, so the peer population storing movie j should be $N\eta_j$. Without any random variations, server bandwidth utilization is 0, and this optimal solution is achieved with $L = 1$.

Given the random demand model, even if the popularity is stationary, the problem is rather complicated. The server load B can no longer be simply expressed in terms of the replication allocation Q_i. However, we can introduce a random algorithm to only generate replication allocations $\{S_j\}$ (or $\{Q_i\}$) that balance the load for all movies from an expectation basis, and try to analyze the algorithm. We found such an algorithm that will be referred to as *Random with Load Balancing* (RLB) is a centralized[2] algorithm, not necessarily practical for implementation. Yet, its result can be considered as a target that any desirable replication algorithm tries to achieve; furthermore, its performance can be characterized in terms of a reasonably tight upper and lower bound[3], under the

[1]The general case without $N \gg K$ is also interesting, and will be addressed in our future work.

[2]A distributed version will be introduced in a later section.

[3]Later, using simulation, we will verify that RLB indeed satisfies the performance bounds.

perfect fair sharing assumption. In section V, an adaptive RLB algorithm more suitable for practical implementation will be described.

Algorithm 1 Random with Load Balancing Assignment

```
1: for $j = 1$ to $K$ do
2:     $B_j = 0$
3: end for
4: for $i = 1$ to N do
5:     Peer $i$ randomly selects $L$ movies from the movie set and puts the id of each movie into set $Q_i$;
6:     $\lambda_i = N \sum_{j \in Q_i} \eta_j$
7:     for $\forall j \in Q_i$ do
8:         $B_j = B_j + \frac{U_i}{\lambda_i}$, for homogeneous, $U_i = 1$
9:         if $B_j \geq 1$ then
10:            Never select movie $j$ any more
11:        end if
12:    end for
13: end for
```

The physical meaning of B_j is the expected received bandwidth for peers watching movie j. In tne next section, our analysis will derive the steady state value of B_j in terms of demand and how movies are replication[4]. If L is equal to 1, the movie allocation result of RLB is the same with the deterministic demand case. Although RLB's performance becomes much more complicated to analyze when L is greater than 1, it is possible to derive closed-form bounds, to be detailed in the next section. These bounds capture the capabilities of all load balancing replication algorithms as well, thus characterizing the generic properties of such P2P VoD streaming systems.

IV. Analysis

In this section, we analyze the performance bounds of RLB for both homogeneous and heterogeneous peer populations.

A. Stationary Demand and Homogeneous Peers

Based on our model, the number of requests at any peer i is a random variable of Binomial distribution, with parameter $p_i = \sum_{j \in Q_i} \eta_j$. For large N, the Binomial distribution can be approximated by the Poisson distribution to simplify the analysis. The probability that peer i receives k requests is:

$$\begin{aligned} Pr_i(Req = k) &= \frac{\lambda_i^k}{k!} e^{-\lambda_i} \ , \\ \lambda_i &= N \times \sum_{j \in Q_i} \eta_j \ . \end{aligned}$$

Let the bandwidth from provider i allocated to a particular peer watching movie j be a random variable $X_j(i)$ where $i \in S_j$. Similarly $X_j(k), k \in S_j$ is a random variable denoting the bandwidth from provider k. The correlation between $X_j(i)$ and $X_j(k)$, $i, k \in S_j$ is denoted $r_j(i,k)$. The expected value and variance of the random variable $X_j(i)$ is:

$$E[X_j(i)] = \frac{1}{\lambda_i}, \text{ and } Var[X_j(i)] \approx \frac{1}{\lambda_i^3}. \tag{1}$$

[4]A random variable X_j will be defined, but $E[X_j[$ is B_j.

The proof is in appendix.

The aggregate bandwidth that peers watching movie j can get from all other peers is $X_j = \sum_{i \in S_j} X_j(i)$. Whenever X_j is less than 1, some peer(s) watching movie j has to seek help from server. Assuming the expected value of X_j is $E[X_j] = 1$, it does not mean peers watching movie j never access the server. The probability that peers seek help from sever depends on the variance of X_j.[5] The variance of X_j denoted by σ_j^2 is determined by the variance of each individual $X_j(i)$ denoted by $\sigma_{j,i}^2$ and the correlation among these $X_j(i)$ for different i. It is difficult to derive the exact relationship between the variance of X_j and correlation $r_j(i,k)$ from peer i and k. But we can derive the upper and lower performance bound by assuming specific correlation values. The higher correlation among these $X_j(i)$, the higher variance of the aggregate bandwidth X_j, which is not expected. The variance can be expressed in terms of the correlation $r_j(i,k)$ as:

$$\begin{aligned} \sigma_j^2 &= \sum_{i \in S_j} Var[X_j(i)] + \sum_{i,k \in S_j \& i \neq k} 2\sigma_{j,i}\sigma_{j,k} r_j(i,k) \\ &\leq (\sum_{i \in S_j} \frac{1}{\lambda_i^{\frac{3}{2}}})^2. \end{aligned} \tag{2}$$

The bound is achieved when $r_j(i,k) = 1$ for all pairs of peers i and k. This is for a single movie. For multiple movies, we use the weighted average variance of all movies in the whole system, which is defined as follows:

$$\sum_{j=1}^{K} \eta_j \sigma_j^2 = \sum_{j=1}^{K} \eta_j (\sum_{i \in S_j} Var[X_j(i)] + \sum_{i \neq k} 2\sigma_{j,i}\sigma_{j,k} r_j(i,k)). \tag{3}$$

This is a key expression - minimizing this weighted variance is equivalent to minimizing the server bandwidth.

Now, our objective is to analyze weighted variance of all movies in the system. If a single movie were to optimize its own playback performance, it would replicate itself at as many peers as possible. Therefore, we need some constraints to restrict the allocation of the limited bandwidth and storage resources. The following condition ensures the allocated expected bandwidth for each peer is equal to the playback rate, and no more. If X_j is less than the playback rate, sever load is required. In our model, the server load is optimized by minimizing the variance of X_j.

$$\sum_{i \in S_j} E[X_j(i)] = 1, \text{ and } \sum_{i \in S_j} \frac{1}{\lambda_i} = 1. \tag{4}$$

On the other hand, the total number of replicas for all movies should be NL, because each peer stores L movies, which gives the following conditions:

$$\sum_{j=1}^{K} \sum_{i \in S_j} \frac{1}{\lambda_i} = K$$

[5]If each peer maintains a buffer, the performance can be improved, because when $X_j > 1$, peers can pre-store some content which is used when $X_j < 1$. By making the fluid assumption, no buffering is included in our model.

Each peer stores exactly L movies, which means $\frac{1}{\lambda_i}$ appears exactly L times for any peer i, therefore the condition can be rewritten as:

$$\sum_{i=1}^{N} \frac{1}{\lambda_i} \;=\; \frac{K}{L}. \tag{5}$$

The RLB algorithm satisfying both condition EQ.(4) and EQ.(5) is balanced on the expected resource allocated to different movies, and satisfies the storage constraints.

The performance of the RLB algorithm is therefore given by EQ.(3). The complicating factor (which cannot be easily derived) is the correlation $r_j(i,k)$, which will affect the variance of the aggregate bandwidth allocated for peers watching movie j, i.e. $Var[X_j]$. The worst case, when the variance attains its maximum value, is reached when the correlation $r_j(i,k)$ is equal to 1 for any movie j. In that case, EQ.(3) then becomes:

$$\sum_{j=1}^{K} \eta_j \sigma_j^2 \;=\; \frac{K}{NL}. \tag{6}$$

The proof is in the appendix. The consumed server bandwidth is derived in closed form below. The best case is achieved when the correlation is minimized. It is impossible to achieve minimized correlation -1 between any two pairs of random allocated bandwidth from different peers because there is at least one common movie for peers $i,k \in S_j$. So we consider $r_j(i,k)=0$ as the correlation that achieves the upper bound.[6] If $r_j(i,k)$ is equal to 0, the weighted variance is:

$$\sum_{j=1}^{K} \eta_j \sigma_j^2 \;=\; \frac{1}{N}\left(\sum_{i=1}^{N} \frac{1}{\lambda_i^2}\right) \;\geq\; \left(\frac{K}{NL}\right)^2. \tag{7}$$

The proof is in the appendix. When $\lambda_1 = \lambda_2 = \cdots = \lambda_N$, the variance achieves the minimum value. Therefore, the performance upper bound is the case when all peers are assigned the same load and these movies are randomly assigned to different peers to minimize correlation of $r_j(i,k)$. The consumed server bandwidth will be derived below as well.

The sever load can be written as the following optimization problem with constraint EQ.(4) and EQ.(5):

$$\begin{aligned} B &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{K}\eta_j \sum_{X_j} |X_j - 1| Pr(X_j) \\ &\leq \frac{1}{2}\sqrt{\frac{NK}{L}} = \frac{1}{2}\frac{N}{\sqrt{M}} \end{aligned} \tag{8}$$

The wastage bandwidth is equal to the server load, because the server load is caused by wastage due to poor allocation. Therefore, there is a factor $\frac{1}{2}$ in EQ.(8). The detailed proof is in the appendix. The condition to achieve the bound requires that $|X_j - 1|$ is a constant. Actually, it is impossible to satisfy the condition for a random variable X_j. We can get a tighter bound by optimizing the correlation of $r_j(i,k)$ in the following discussion.

[6]We conjecture that this is true. A formal proof is for future work.

To derive the worst case server bandwidth, we construct a scenario when the correlation is 1. Let the peers be partitioned into different clusters and peers in each cluster store the same movie set. Assume K can be divided by L, and each cluster randomly selects L movies. All peers in the same cluster will store exactly the same L movies. In this case, the received bandwidth from different peers will have a perfect correlation 1, because all peers will have exactly the same number of requests. The expected requests for a peer i is $\lambda_i = N\sum_{j\in Q_i}\eta_j$. Therefore, the peer population in a cluster with peer i should be $N\sum_{j\in Q_i}\eta_j$. The required server bandwidth in this case can be analyzed and calculated exactly, as:

$$B \;=\; \frac{1}{\sqrt{2\pi}}\sqrt{\frac{NK}{L}} \;=\; \frac{1}{\sqrt{2\pi}}\frac{N}{\sqrt{M}}. \tag{9}$$

The proof is in the appendix. Normally we expect RLB to achieve better results than this worse case performance.

The upper bound can be achieved by assuming $r_j(i,k)=0$. The consumed server bandwidth is:

$$B \;=\; \frac{1}{\sqrt{2\pi}}\frac{K}{L} \;=\; \frac{1}{\sqrt{2\pi}}\frac{N}{M}. \tag{10}$$

EQ.(9) and EQ.(10) can be viewed as a basic *law* of what replication can do. This result can also be used as a rule of thumb to determine storage space, or number of movies, based on peer population in a p2p VoD system. It also gives us the following insight: although smart replication algorithms can reduce server load to close to zero under deterministic demand; but under random (stationary) demand, the best replication algorithm must not only balance the load among different movies, but also reduce the correlation of a movie's demand on different peers at the same time.

B. Stationary Demand and Heterogeneous Peers

It is easy to extend our model to the heterogeneous peers case by letting the upload capacity of peer i be U_i. The movie allocation algorithm is applicable in heterogeneous case. Because of the fluid assumption, each peer can equally divide U_i bandwidth among all its requests. EQ.(1) is rewritten as:

$$E[X_j(i)] \;=\; \frac{U_i}{\lambda_i}, \text{ and } Var[X_j(i)] \;=\; \frac{U_i^2}{\lambda_i^3}. \tag{11}$$

Proposition *1:* The heterogeneous case shares the same lower bound with the homogeneous case.

The proof is very similar to the proof of equation EQ.(9). The worst case is still the case when correlation is 1. The worst performance can again be constructed by dividing peers into clusters. The difference is the peer population of each cluster. Let C_t denote the peer set of cluster t. There are $\frac{K}{L}$ clusters. In the heterogeneous case, the peer population for cluster t should satisfy the condition EQ.(4), i.e. $\sum_{i\in C_t}\frac{U_i}{N\sum_{j\in Q_i}\eta_j} = 1$. The peer population of cluster t has no closed form, but this will not affect the performance of the RLB algorithm.

Proposition *2:* The heterogeneous case shares the same upper bound with the homogeneous case.

The proof is similar to the proof of EQ.(10), but the condition to achieve the best performance is slightly different. For the heterogeneous case, the weighted average variance of all movies becomes:

$$\sum_{j=1}^{K}\eta_j\sigma_j^2 \;=\; \frac{1}{N}(\sum_{i=1}^{N}\frac{U_i^2}{\lambda_i^2}) \;\geq\; (\frac{K}{NL})^2. \qquad (12)$$

With the constraint $\sum_{i=1}^{N}\frac{U_i}{\lambda_i}=\frac{K}{L}$, the condition to achieve minimum variance is $\frac{U_1}{\lambda_1}=\cdots=\frac{U_N}{\lambda_N}$. Actually, the condition is hard to satisfy in real systems. The important result is that the lower bound for the heterogeneous case is no worse than the homogeneous case.

V. Adaptive Algorithm

So far, we have not discussed how the peers do movie replication based on the watched movies. In practice, it is expensive to push movies to every peer by our RLB algorithm. In this section, we propose and study a distributed adaptive RLB (ARLB) algorithm, which do movie replication based on the watched movies. After a peer i watched a movie j, the peer runs the Adaptive RLB replication algorithm. The pseudo code is in Alg. 2. Note, $x^+ = x$ if $x > 0$, else 0. The physical

Algorithm 2 Adaptive RLB Assignment

```
1:  if $|Q_i| < L$ then
2:      Store movie $j$ on peer $i$; Return;
3:  end if
4:  if $B_j \geq 1$ then
5:      Do not do replication
6:  else
7:      $r = argmax_{k\in Q_i}\eta_k(B_k - 1)$
8:      $\lambda_B = N\sum_{k\in Q_i}\eta_k$
9:      $\lambda_E = \lambda_B + (\eta_j - \eta_r)N$
10:     for $\forall k \in Q_i$ do
11:         if $k \neq r$ then
12:             $B_{kE} = B_k + \frac{1}{\lambda_E} - \frac{1}{\lambda_B}$
13:         else
14:             $B_{kE} = B_k - \frac{1}{\lambda_B}$
15:         end if
16:     end for
17:     $B_{jE} = B_j + \frac{1}{\lambda_E}$
18:     $GAP_B = \sum_{k\in Q_i}\eta_k(1 - B_k)^+ + \eta_j(1 - B_j)^+$
19:     $GAP_E = \sum_{k\in Q_i}\eta_k(1 - B_{kE})^+ + \eta_j(1 - B_{jE})^+$
20:     if $GAP_E < GAP_B$ then
21:         replace movie $r$ with movie $j$
22:         delete $r$ from $Q_i$ and add $j$ into $Q_i$
23:     end if
24: end if
```

meaning of B_j is the same as that in the RLB algorithm. The objective of ARLB is to achieve balanced bandwidth for all movies, i.e. $B_j = E[X_j] = 1$ for any movie j. The objective is easily achieved by the centralized algorithm in RLB. ARLB is an adaptive version of RLB. The peer calculates the weighted gap between B_j and required playback rate 1 before and after movie replication. If the gap can be reduced with replicating the new movie, the peer will do that. This algorithm is evaluated by simulation, and is compared with other adaptive algorithms, including Random replacement, LFR, and FIFO. For implementation, it is necessary to set up a central server to collect and provide the global information of B_j for all movies. Each peer is required to periodically update B_j of its stored movies (including the one currently being viewed) and report them to the central server. In practice, this will incur some overheads.

VI. Simulation

In this section, we present the simulation results that are organized as two parts.

In the first part, the experiments are based on the stationary model and static replication. The movies stored at each peer are assigned at the beginning, and stay unchanged. Our experiments validate our model, and the lower- and -upper bound of the RLB algorithm. We include the equal copy number (ECN) algorithm and the Proportional algorithm as benchmark. ECN tries to make equal number of copies for each movie under any movie popularity distribution, while, proportional algorithm tries to make the number of copies for movies proportional to their popularity, a heuristic algorithm described in [1]. We also study sensitivity on parameters such as N, K, L, movie popularity distribution and uplink capacity distribution.

In the second part, performance of four adaptive replication algorithms including FIFO, LFR (least frequently requested), Random (random replacement) and our proposal, Adaptive Random Load Balance (ARLB), are compared through simulations under stationary demand model.

A. Stationary demand and static replication assignment.

a) Model validation under homogeneous settings: We first validate our statistical analysis results. In this simulation, we apply stationary demand model with evenly distributed movie popularity ($\eta_j = \frac{1}{K}, \forall j$) and homogeneous peer uplink capacity ($U_i = 1, \forall i$). The total simulation duration is 1500 timeslots and $N = 10000$. The viewing duration is set to be uniformly distributed in $[20, 40]$ timeslots. After finishing viewing a movie, each peer will make another movie selection based on the given movie popularity independently and memorylessly.

In order to keep the values of upper- and lower-bound unchanged, we fix the ratio $K/L = 50$, but change $L = 1, 2, 4, 8, 16, 32, 48, 64, 80$. The results are shown in Fig 1.

We can observe that: 1) The average server load decreases when L is increased; 2) The average server load of RLB is strictly bounded; 3) The lower-bound is achieved at $L = 1$ while the upper-bound is approached when L becomes much larger.

b) Sensitivity analysis on configuration parameters: The sensitivity of the replication algorithms to different system parameters are shown in Fig.2.

In Fig.2(a), we fix $N = 10000$, $K = 200$, $L = 4$ and homogenous peer uplink capacity, and repeat the simulations

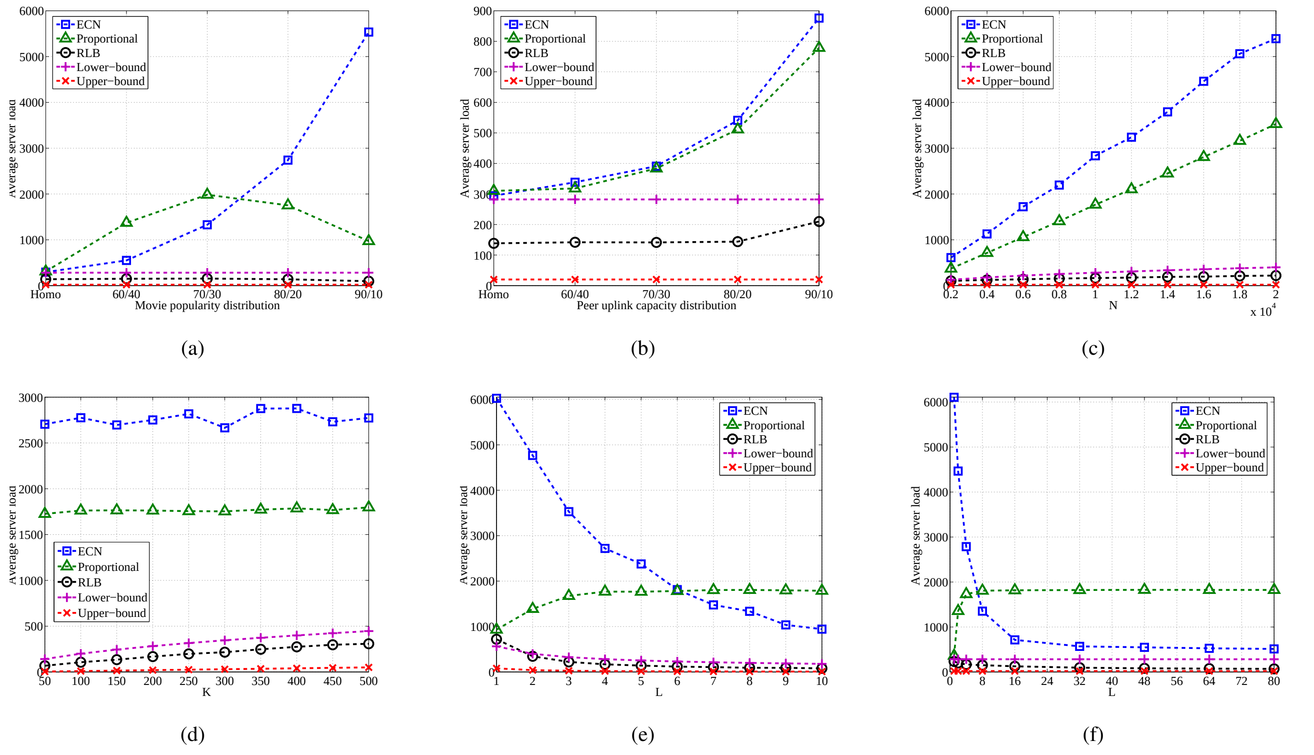

Fig. 2. Sensitivity analysis on configuration parameters.

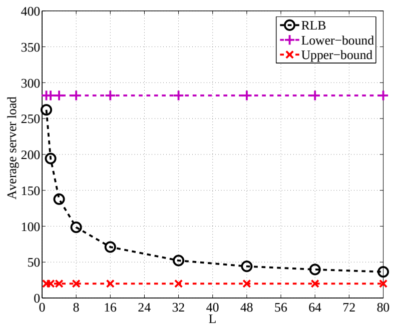

Fig. 1. Validate statistical model analysis and achieve theoretical upper- and lower- bounds.

for five times under five different movie popularity distributions, where $X/(100-X)$ means that X percent of the movies equally share the $(100-X)$ percent of the total request rate. Take the 80/20 as an example, if $K = 200$, then 40 movies have $\eta_j = 0.8/40 = 0.02$ while the remaining 160 movies have $\eta'_j = 0.2/160 = 0.00125$. In this case, we consider the 40 movies as *hot* movies while the remaining as *cold* movies. The results are shown in Fig.2(a).

In a similar way, we plot the results of changing uplink capacity distribution in Fig.2(b).

Under 80/20 movie popularity distribution and 80/20 uplink capacity distribution, we did the following experiments:

1. Changing N from 2000 to 20000 with $K = 200$ and $L = 4$ fixed, shown in Fig.2(c).
2. Changing K from 50 to 500 with $N = 10000$ and $L = 4$ fixed, shown in Fig.2(d).
3. Changing L from 1 to 10 with $N = 10000$ and $K = 200$ fixed, shown in Fig.2(e).
4. Changing $L = 1, 2, 4, 8, 16, 32, 48, 64, 80$ with $N = 10000$ and ratio of $K/L = 50$ fixed, shown in Fig.2(f).

As shown in Fig.2, we can observe in all the six cases that: 1) RLB performs much better than the other two algorithms. 2) The average server load of RLB is strictly within the upper- and lower- analytical bounds, as expected.

B. Evaluate adaptive replication algorithms

In this part, we evaluate four adaptive replication algorithms under the stationary demand model. In the adaptive replication, each peer will dynamically adjust its local stored movies according to what it has viewed. The simulation results are shown in Fig.3.

Under 80/20 movie popularity distribution and 80/20 uplink capacity distribution, we conducted four simulation experiments where equal copy number (ECN) assignment algorithm is applied in the initialization.

1. Changing N from 2000 to 20000 with $K = 200$ and $L = 4$ fixed, shown in Fig.3(a).
2. Changing K from 50 to 500 with $N = 10000$ and $L = 4$ fixed, shown in Fig.3(b).
3. Changing L from 1 to 10 with $N = 10000$ and $K = 200$ fixed, shown in Fig.3(c).

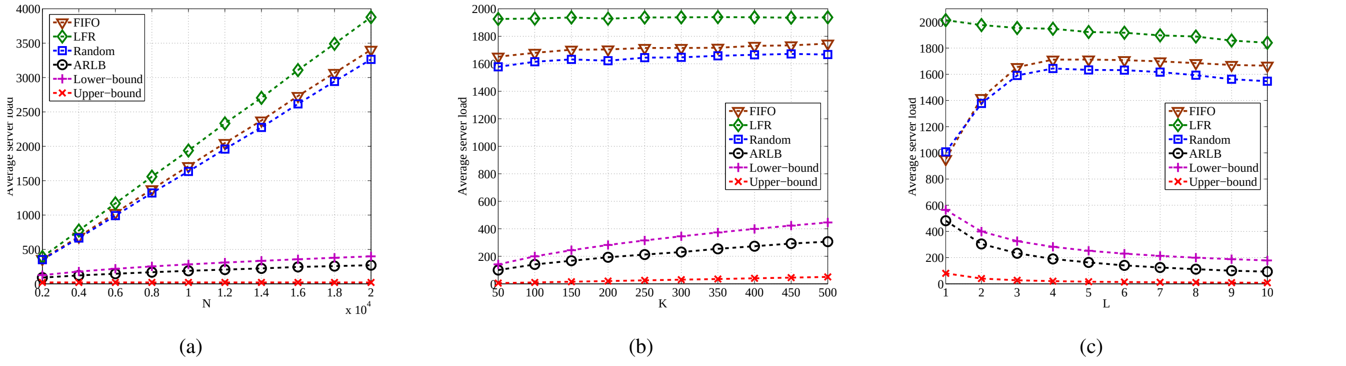
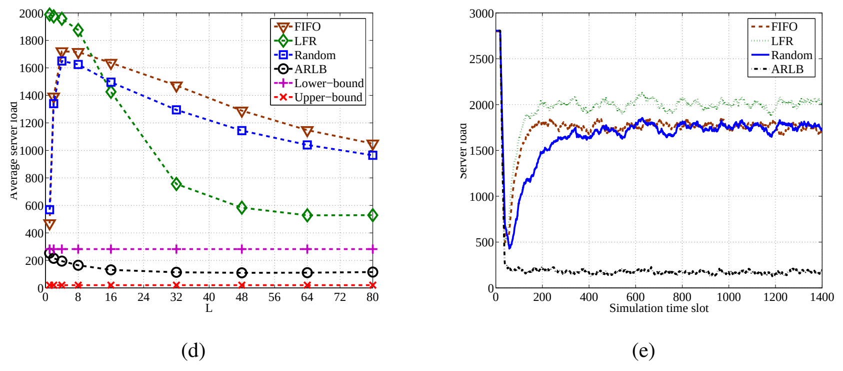

Fig. 3. Performance evaluation on dynamic replication algorithms. Under stationary movie popularity, sensitivity analysis on configuration parameters: (a) N, (b) K, (c) L and (d) fixed ratio of K/L. (e) The detailed server load value in every timeslot, with $N = 10000, K = 200, L = 4$.

4. Changing $L = 1, 2, 4, 8, 16, 32, 48, 64, 80$ with $N = 10000$ and ratio of $K/L = 50$ fixed, shown in Fig.3(d).

As shown in Fig.3(a), 3(b), 3(c) and 3(d), we can observe in all the four cases that: 1) ARLB performs much better than FIFO, LFR and Random replacement algorithms. 2) The average server load of ARLB is still bounded by upper- and lower-bounds.

Fig.3(e) plots the server load over the duration of the simulation for each algorithm. The parameters for this scenario are $N = 10000$, $K = 200$ and $L = 4$. As shown in Fig.3(e), the result is very interesting. There are mainly three phases: 1) All four algorithms incur a very high server load (around 2800), which is because all of them apply equal copy number (ECN) assignment algorithm in the initialization. 2) After that, all four algorithms quickly decrease to a low server load point (around 500) with about the same speed. 3) Finally, after reaching the valley point, FIFO, LFR and Random replacement gradually increase (at different slope) to a "stable" value and oscillate around it until the end. However, for the ARLB case, the server load does not increase but remain at the valley point (around 200). We can explain this as follows. Since the movie popularity is 80/20, there are hot movies ($\eta_j = 0.02$) and cold movies ($\eta_j = 0.00125$). The ECN initial assignment causes the initial shortage of hot movie copies in the peers' local storage. This is why the four curves start at a high server load point. Hot movies are more likely chosen and stored locally than cold movies by all these algorithms. Quickly, the number of hot movie copies in peers' local storage increases a lot so that the server load decreases rapidly. After the number of the hot movie copies reaches a critical value (the valley point of the server load), it continues increasing for FIFO, LFR and Random replacement. This behavior causes imbalance of the load and resource, so the server loads for these three algorithms increase after the valley point. However, for ARLB, it automatically balance hot and cold movie copies in the local storage so that the server load remains at the minimum.

VII. Related Works

The following related works are either influential or closely related to our work. We discuss them one by one.

In [1], the authors studied a (probably the first) large-scale, deployed P2P VoD system. This is an architecture and measurement paper. It points out that P2P replication is a key design issue. The replication algorithm used in this system is the Proportional algorithm (we compared to it in our simulation). In [3], the authors propose a set-top box based P2P system as a form of data center, albeit a highly distributed one with central management, to be used for applications including P2P VoD. They discussed many architectural issues in such designs. These two papers provide good motivations for our work.

The closest work to ours are [7], [8]. These papers are about a multi-channel P2P live streaming system with service-viewing decoupling, which means peers may serve other peers with content they are not viewing. This setting is actually very similar to the multi-movie VoD system we are considering. In multi-channel streaming, a peer does not contribute storage, but only shares the content it buffers. This means decoupling will always incur some additional overhead (to download content purely for serving others). But decoupling is a good idea because it can alleviate the problem caused by frequent channel jumps by peers. In P2P VoD, decoupling is automatic, incurring no (real-time) bandwidth overheads. While there are some similarities between our models, the key breakthrough in our work is the assumption of the *Perfect Fair-Sharing* service model that allows us to derive closed-form solutions for the performance bounds and other properties. In [9], the authors created a fluid model to analyze a P2P VoD system with dynamic peer arrivals and departures. But this study considered only one movie, and did not model the user behavior, which is an important ingredient of current P2P VoD systems and is studied in our model. Paper [10] studied movie replication strategy in P2P VoD systems. However, the proposed replication strategy is exponential time complexity. The required server bandwidth is not reduced significantly compared with other strategies such as LRU, and LFU.

The following are some other recent studies of various approaches to P2P VoD. In [11], the author proposed an architecture for P2P VoD system, including an ARIMA module to predict the popularity of each movie. Based on the prediction, the author proposed a heuristic algorithm to do replication. In [12], the author designed an architecture for P2P VoD system with rateless coding. The movie is divided into segments. Each segment is encoded by rateless code. If a peer wants to download a segment, the peer can seek help from any neighbors with partial content, because rateless code is a random coding scheme. In [13], the authors proposed a similar idea based on using network coding.

VIII. Conclusion

In this paper, we propose a service model (perfect fair sharing) and a stationary statistical demand model for P2P VoD, so that a class of near optimal movie replication algorithms (RLB) can be analyzed. The result is a general scaling law that P2P VoD has to satisfy, in terms of how peer population, movie number, and replication capacity at each peer. We also describe a more adaptive version of RLB that is distributed, and more practical for implementation. We show that the adaptive RLB can out-perform many other well-known algorithms, though this algorithm does still require some global information. For future work, we will model dynamic movie popularity, and study replication algorithm under that scenario.

References

[1] Y. Huang, T. Z. J. Fu, D. M. Chiu, J. C. S. Lui, and C. Huang, "Challenges, design and analysis of a large-scale p2p vod system," in *Proc. of ACM Sigcomm*, 2008.

[2] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "Coolstreaming/donet: A data-driven overlay network for efficient live media streaming," in *Proc. of IEEE Infocom*, 2005.

[3] N. Laoutaris, P. Rodriguez, and L. Massoulie, "Echos: Edge capacity hosting overlays of nano data centers," in *CCR*, 2008.

[4] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *Proc. of ACM Sigcomm*, 2002.

[5] S. Tewari and L. Kleinrock, "Proportional replication in peer-to-peer networks," in *Proc. of IEEE Infocom*, 2006.

[6] J. Kangasharju, K. W. Ross, and D. A. Turner, "Optimizing file availabiity in peer-to-peer content distribution," in *Proc. of IEEE Infocom*, 2007.

[7] D. Wu, Y. Liu, and K. W.Ross, "Queuing network models for multi-channel live streaming systems," in *Proc. of IEEE Infocom*, 2009.

[8] D. Wu, C. Liang, Y. Liu, and K. Ross, "View-upload decoupling: A redesign of multi-channel p2p video systems," in *Proc. of IEEE Infocom*, 2009.

[9] Y. Lu, J. Mol, F. Kuipers, and P. V. Mieghem, "Analytical model for mesh-based p2pvod," in *IEEE International Symposium on Multimedia*, 2008.

[10] J. Wu and B. Li, "Keep cache replacement simple in peer-assisted vod systems," in *Proc. of IEEE Infocom*, 2009.

[11] C. Loeser, G. Schomaker, A. Brinkmann, M. Vodisek, and M. Heidebuer, "Content distribution in heterogenous video-on-demand p2p networks with arima forecasts," in *LECTURE NOTES IN COMPUTER SCIENCE*, 2005.

[12] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Valleo, "Push-to-peer video-on-demand system: design and evaluation," in *IEEE Journal on Selected Areas in Communications, special issue on Advances in Peer-to-Peer Streaming Systems*, 2007.

[13] E. Kehdi and B. Li, "Incorporating random linear network coding for peer-to-peer network diagnosi," in *Proc. of IEEE Infocom*, 2010.

Appendix

Proof of Eq. (1): A particular peer wants to watch a particular movie j and tries to get bandwidth from peer i. The obtained bandwidth depends on the total requests on peer i. The received requests still follow Poisson distribution. Without the particular peer, we assume the requests caused by left peers still is a Poisson distribution with the same expected value λ_i. The error due to approximation can be ignored when $N \gg 1$.

$$E[X_j(i)] = \sum_{k=0}^{+\infty} \frac{1}{k+1} \frac{\lambda_i^k e^{-\lambda_i}}{k!} = \frac{1}{\lambda_i}(1 - e^{-\lambda_i}) \approx \frac{1}{\lambda_i},$$

$$\begin{aligned} Var[X_j(i)] &= E[X_j(i)^2] - (E[X_j(i)])^2 \\ &= \sum_{k=0}^{+\infty} \frac{1}{(k+1)^2} \frac{\lambda_i^k e^{-\lambda_i}}{k!} - \frac{1}{\lambda_i^2} \\ &= \frac{1}{\lambda_i^3} + \sum_{i=0}^{+\infty} \frac{2}{(k+3)!(k+1)} \frac{\lambda_i^k e^{-\lambda_i}}{k!} + O(e^{-\lambda_i}) \\ &= \frac{1}{\lambda_i^3} + O(\frac{1}{\lambda_i^4}) \approx \frac{1}{\lambda_i^3}. \end{aligned}$$

■

Proof of EQ.(6): When there is a perfect correlation between any two $X_j(i)$ and $X_j(k)$, and the peers are divided into clusters and each cluster store the same movie set, the peer population of cluster with peer i and including movie j is equal to $|S_j| = N\sum_{j\in Q_i} \eta_j = \lambda_i$. Therefore, the variance of $\sum_{i\in S_j} X_j(i)$ is:

$$\sigma_j^2 = Var[\sum_{i\in S_j} X_j(i)] = \frac{|S_j|^2}{\lambda_i^3} = \frac{1}{|S_j|}.$$

The weighted total variance is:

$$\sum_{j=1}^{K} \eta_j \sigma_j^2 = \sum_{j=1}^{K} \frac{\eta_j}{|S_j|} = \sum_{i=1}^{K/L} \frac{\sum_{j\in Q_i} N\eta_i}{N|S_j|} = \frac{K}{NL}.$$

■

Proof of EQ.(7): When the correlation is 0, the variance of $\sum_{i\in S_j} X_j(i)$ is:

$$\sigma_j^2 = Var[\sum_{i\in S_j} X_j(i)] = \sum_{i\in S_j} \frac{1}{\lambda_i^3}.$$

The weighted total variance is:

$$\sum_{j=1}^{K} \eta_j \sigma_j^2 = \sum_{j=1}^{K} \sum_{i\in S_j} \frac{\eta_j}{\lambda_j^3} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\lambda_i^2} \geq (\frac{K}{NL})^2.$$

From the constraint EQ.(5), the inequality achieves minimum value, when $\lambda_1 = \lambda_2 = ... = \lambda_N$. Therefore the best case is that all peers are assigned the same load, and these movies are randomly distributed so that the correlation among different peers is minimized. ■

Proof of EQ.(8): The server load is rewritten as follow and the bound is derived from Cauchy-Schwarz inequality:

$$\begin{aligned} B &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{K}\sqrt{\eta_j}\sum_{X_j}\sqrt{\eta_j}|X_j - 1|\sqrt{Pr(X_j)}\sqrt{Pr(X_j)} \\ &< \frac{1}{2}\sum_{i=1}^{N}\sqrt{\sum_{j=1}^{K}\eta_j\sum_{X_j}Pr(X_j)}\sqrt{\sum_{j=1}^{K}\eta_j(X_j-1)^2 Pr(X_j)} \\ &= \frac{1}{2}N\sqrt{\sum_{j=1}^{K}\eta_j\sigma_j^2} \leq \frac{1}{2}\sqrt{\frac{NK}{L}} \end{aligned}$$

Here, we use the inequality $\sqrt{\sum_{j=1}^{K}\eta_j\sigma_j^2} \leq \frac{K}{NL}$ which is proved in EQ.6. The condition to achieve the bound is that $|X_j - 1|$ is equal to a constant number. This can not be satisfied in our model because X_j is a random variable with many possible values. ■

Proof of EQ.(9): From above analysis, in the worst case, the peers are divided into clusters and in each cluster, all peers store exactly the same movie set. Actually, a cluster is like a super server and the upload capacity is the sum of the upload capacity of all its members. In our strategy, the aggregate upload capacity of a group is equal to the expected request received by a peer in the group, i.e. $\sum_{i\in C_t} U_i = \lambda_{C_t}$. There are $\frac{K}{L}$ clusters. If the total requests are less than the aggregate upload capacity λ_{C_t}, all requests can be satisfied. If there is more than λ_{C_t} requests, the sever has to provide the unserved bandwidth. The number of requests is a Poisson random variable. The expected allocated bandwidth for a peers watching any movie in cluster C_t is:

$$\begin{aligned} EB_{C_t} &= 1 \times Pr(rq < \lambda_{C_t}) + \sum_{k=\lambda_{C_t}}^{+\infty}\frac{\lambda_{C_t}}{k+1}Pr(rq = k) \\ &= \Pr(rq < \lambda_{C_t}) + Pr(rq > \lambda_{C_t}) \\ &= 1 - \frac{\lambda_{C_t}^{\lambda_{C_t}}}{\lambda_{C_t}!}e^{-\lambda_{C_t}}. \end{aligned}$$

According to Stirling's Approximation, i.e. $\lambda! \approx \sqrt{2\pi\lambda}(\frac{\lambda}{e})^{\lambda}$, we substitute it back to the above equation and the expected bandwidth obtained becomes:

$$EB_{C_t} = 1 - \frac{1}{\sqrt{2\pi\lambda_{C_t}}}.$$

The weighted expected unserved bandwidth for each peer in the system is:

$$\sum_{t=1}^{K/L}\frac{\lambda_{C_t}}{N}(1 - EB_{C_t}) = \frac{1}{N\sqrt{2\pi}}(\sum_{t=1}^{K/L}\sqrt{\lambda_{C_t}}),$$

where constraint is $\sum_{t=1}^{K/L}\frac{\lambda_{C_t}}{N} = 1$. It is not hard to find the upper bound of the above formula, $\frac{1}{\sqrt{2\pi}}\sqrt{\frac{K}{NL}}$. When $\lambda_{C_1} = ... = \lambda_{C_{K/L}}$, the system achieve the worst case. Therefore, the required server bandwidth is $B = \frac{1}{\sqrt{2\pi}}\sqrt{\frac{NK}{L}} = \frac{1}{\sqrt{2\pi}}\frac{N}{\sqrt{M}}$. ■

Proof of EQ.(10): The best performance is achieved when all peers have the same expected load $\lambda_1 = \lambda_2 = ... = \lambda_N$ and there is no correlation among the allocated bandwidth from different peers. That means $X_j(i)$ for $i \in S_j$ are identical independent distribution. So the aggregate bandwidth X_j follows normal distribution with mean 1. Because of the symmetry property of Normal distribution, the probability that the aggregate bandwidth is less than mean is $\frac{1}{2}$. The variance is the sum of individual variance. From EQ.(1), the variance for the bandwidth from one peer is $\frac{1}{\lambda_i^3}$. Letting $f(X_i)$ denote the probability distribution of the aggregate bandwidth X_j, the expected aggregate bandwidth for peer i can be derived as follows:

$$\begin{aligned} EB_j &= 1 \times \frac{1}{2} + \int_{-\infty}^{1} xf(x)dx \\ &= \frac{1}{2} + \int_{-\infty}^{1}\frac{x}{\sqrt{2\pi\sigma_i^2}}e^{-\frac{(x-u)^2}{2\sigma_i^2}}dx = 1 - \frac{\sigma_i}{\sqrt{2\pi}}. \end{aligned}$$

The total consumed server bandwidth is:

$$B = \sum_{i=1}^{N}(1 - EB_j) = \frac{1}{\sqrt{2\pi}}(\sum_{i=1}^{N}\sigma_i).$$

Now the task is to find the value of σ_i. From the constraints EQ.(4), EQ. (5) and the condition $\lambda_1 = \lambda_2 = ... = \lambda_N$, we have:

$$\lambda_i = \frac{K}{NL}.$$

The number of copies for each movie is equal to $\lambda_i = \frac{NL}{K}$. The variance of aggregate bandwidth is the sum of individual variance because of independent assumption. So $\sigma_i = \sqrt{\frac{NL}{K}\frac{1}{\lambda_i^3}} = \frac{K}{NL}$. The consumed server bandwidth is:

$$B = \frac{1}{\sqrt{2\pi}}\frac{K}{L} = \frac{1}{\sqrt{2\pi}}\frac{N}{M}.$$

■