A Hierarchical Network Storage Architecture for Video-on-Demand Services

Yuan-Cheng Lai Ying-Dar Lin Horng-Zhu Lai
Department of Computer and Information Science,
National Chiao Tung University
Hsinchu, Taiwan

Abstract: *Recent advances in Cable TV networks and multimedia technologies open the possibilities for network/service/content providers to offer residential customers with video-on-demand services. However, the mass storage system in supporting such services demands proper organization and management. In this paper we present a three-level hierarchical network storage architecture for the video-on-demand storage system. At the first-level (Local Service Center, LSC) a limited number of programs with high viewing probabilities are stored; while at the second-level (Local Central Service Center, LCSC) a few programs with second high viewing probabilities are stored. The third-level (Central Service Center, CSC) contains all programs provided in the system. Based on this architecture and the program viewing probability distribution function, we use a minimum-cost function to find out the numbers of programs stored in the two service centers (LSC and LCSC) and numbers of links among these three service centers. We also describe two program reallocation algorithms which swap programs between service centers according to the change in user request patterns.*

1. Introduction

Recent advances in optical transmission, high speed packet switching, data compression, and storage system design will allow network operators to provide residential customers with high bandwidth interactive services such as video-on-demand, video games, and home shopping, entertainment, etc., on a per-user per-session basis in real-time. Users of such services have flexibility in choosing the kinds of information they receive and some control capabilities with the services. In the video-on-demand service, users can control movie presentation during the session. A true VOD system supports full virtual VCR(Video Cassette Recorder) capabilities such as play, forward, reverse, and pause. Video-on-demand seems to be the most attractive service. But it is also one of the most difficult services to provide.

The authors are with the Department of Computer and Information Science, National Chiao Tung University Hsinchu, Taiwan

Various studies have addressed specific issues in the design of a VOD system. The general architecture in designing a VOD system have been investigated by Gelman[1], Woodward[2], Deloddere[3], Sincoskie[4] and Thomas[5]. Several researchers have investigated server storage organizations for supporting VOD systems [6-9].

A general VOD system with a two-level network storage architecture may consist of a program archives connected, through a high speed backbone network, to many local servers. Programs are cached locally and subsequently played back and delivered to the local users via a community network. By adding the two-way signaling capability, we may turn CATV networks to be the most suitable community networks for VOD systems, due to their large bandwidth, low cost, and wide availability. On this premise, we can combine the program providers and CATV operators to offer VOD service for residential customers on the CATV networks. We may also apply the two-level storage architecture to the on CATV network. Thus, in this paper, we present a three-level hierarchical network storage architecture for the VOD storage system.

In this architecture, there are many CSCs(Central Service Centers), which are maintained by program providers, and many CATV operators, all connected by a backbone network. Each CATV operator maintains a LCSC(Local Central Service Center) and a few LSCs(Local Service Centers) for the regions it serves. A LSC has only a limited number of popular programs and may need to download programs from its LCSC or even the CSC if requested programs cannot be found locally. We revise the minimum-cost method developed by Giovanni, Langellotti and Petrini [6] for the three-level architecture to find out the storage capacities in the LCSC and LSC and the number of digital links between the levels.

This paper is organized as follows. Section 2 described our network storage architecture and develops its cost model. The storage requirements at LSCs and LCSCs, ant the numbers of links between LSCs, LCSCs, and CSC are derived. Numerical results are given in section 3 where issues on optimal program allocation influence of program viewing probabilities and cost parameters are examined. Since program viewing probabilities may change over time, we develop program re-allocation algorithms in section 4. Different algorithms are given for LSC/LCSC with fixed or variable storage capacities. Finally section 5 concludes the paper.

2. Network Storage Architecture

In our three-level network storage architecture for video-on-demand services(see figure 1), we use ATM and CATV as the backbone network and community network, respectively. CSCs(Central Service Centers) are located in the ATM network and belong to the program providers, while LCSCs and LSCs are located in the CATV network and belong to the CATV operators. There are several advantages of this architecture:

- **Reduced storage and delivery cost:** According to the user viewing behavior, the CATV operator only has to store a few programs with high viewing probability inside LCSC and LSC, and the system may satisfy most user requests, which reduces both storage cost and delivery cost. This is especially true when the delivery cost to retrieve a program from CSC , through the ATM backbone, is very high.

- **Higher program availability:** Because the programs are distributed to many local servers, the availability and reliability will be increased.

- **Reduced wide area traffic:** Most traffic will be local traffic between customers and LSCs. Traffic between LSC and LCSC, and between LCSC and LSC can be much reduced.

- **Structured management:** It is easier to manage, as each LSC is responsible for the management of its own region and each LCSC is responsible for its community CATV network.

In the system, the service centers containing video programs are layered in the three-level network storage architecture. The requests from the customers are handled by the Local Service Centers(LSC) belonging to the first-level and, if the video program required is not found in LSC, the request is passed to the second level(LCSC, Local Central Service Center). LSC and LCSC are all in the community network. If the video program required is not found in LCSC, the request is further passed to the third level(CSC, Central Service Center) via the up-link channel on the backbone network.

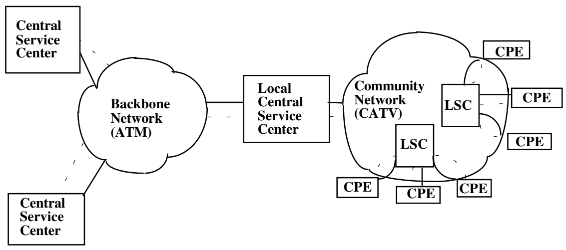

Figure 1 Three-level Architecture for VOD System

System Overview

The components of the system include LSC, LCSC and CSC as we described below:

I. LSC(Local Service Center)

Local Server: The local server handles requests from local users. The local server will plays back the program if the requested program is stored in LSC. If the requested program is stored only in LCSC or CSC, it passes the request to LCSC and then downloads the program from LCSC.

Storage : The programs stored in LSC have high viewing probabilities. Therefore, RAID(Redundant Array of Inexpensive Disks)[10] is very applicable for our requirements.

II. LCSC(Local Central Service Center)

Local central server: The local central server handles the requests from the LSC. If the requested program is stored in LCSC, the selected program is transported from LCSC to LSC. If the selected program is stored only in CSC, the local central server acts as the switch between CSC and LSC.

LCSC does not have playback capability.

Storage: The programs stored in LCSC have the less high viewing probabilities movies. Therefore, RAID or hierarchical storage[7] is suitable for this purpose.

III. CSC(Central Service Center)

Central server: All programs provided in our system are stored in CSC. The central server handles the requests from LCSCs via the backbone network. CSC does not have playback capability.

Mass Storage: The storage capacity must be of a very large scale. It is too expensive to store all programs on RAID. If all program are stored on a tertiary storage device, the system performance will be degraded. Thus storing programs stored on a hierarchical storage structure is suitable for this purpose.

Program Viewing Probability Model

Subsequently, we need to find out the numbers of programs to be stored in LSC and LCSC to optimize the system cost. There are two kinds of cost factors in the overall system cost: delivery cost and storage cost. They depend on the program allocation, but program allocation depends on the program viewing probability, so we have to define the program viewing probability first.

Definition

The program viewing probabilities assuming M programs in the system, can be defined as

$$p_1 \ , \ p_2 \ , \ p_3 \ \cdots \ p_M \quad \textit{where} \quad \sum_i^{M} p_i = 1 \quad \textit{and}$$

$$p_1 > p_2 > p_3 > \cdots > p_M$$

p_1 represents the highest viewing probability and p_M represents the lowest. Since the exact viewing probability distribution is unknown, the following function can be assumed for simplicity:

which means $p_i = \dfrac{(1+\alpha)\alpha^{i}}{\cdots^{M}}$

The parameter α is the ratio of the (i+1)-th and the i-th program viewing probabilities. Figure 2 shows p_i as a function of k for various α values. We can find out that the request distribution becomes highly skewed for small α, and it remains quite uniform for large α.

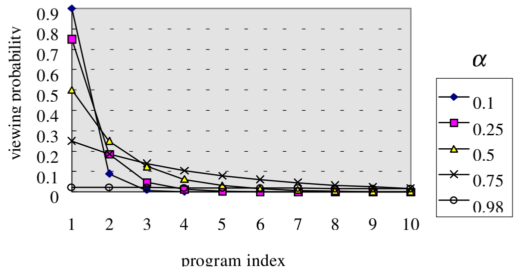

Figure 2 Viewing probability vs. program index for various α

Program Allocation Method

We allocate programs to LSC and LCSC in conformance with the program viewing probability of the system. The program viewing probability can change rapidly by many reasons such as Oscar nominations or holidays. The user's interest is the primary factor anyway. Consequently, we assume that a management system capable of reallocating the most popular programs in LSC and LCSC periodically. We address this problem and present two reallocation algorithms later. The performance of the VOD system is affected by the program assignment

After we have allocated programs to LSC and LCSC, we will consider the program copy assignment problem in server's disks. According to the method described in [8], we have the following two principles:

1. There is no improvement in connection-setup probability when more than one copy of the same program are placed on the same disk.
2. The blocking probability of a customer's viewing request is minimal when each has a uniform distribution of user access probability.

Cost Parameters

We now introduce the minimum cost model. First, we have to define parameters used in the system. Figure 3 illustrates the location of the parameters in the system, the items with underlines are key parameters. The parameters are divided into three categories: storage parameter, network parameter and system parameter. We describe them as follows:

1. Storage Parameter

***SLSC*:** storage capacity, in units of program copies, in the LSC

***SLCSC*:** storage capacity, in units of program copies, in the LCSC

$Cp1$: storage cost of a single program copy in LSC

$Cp2$: storage cost of a single program copy in LCSC

$Vhd1$: maximum number of sessions served at the same time by a single copy of program stored on LSC at the same time.

$Vhd2$: maximum number of sessions served at the same time by a single copy of program stored

2. Network Parameter

$Nlink1$: number of dedicated links between LSC and LCSC

$Nlink2$: number of dedicated links between LCSC and CSC

C_{link1}: cost of a single dedicated link between LSC and LCSC

C_{link2} : cost of a single dedicated link between LCSC and CSC

3. System Parameter

M: total number of distinct programs in the system

U: number of customers connected to a LSC

$k1$: number of distinct programs stored in LSC

$k2$: number of distinct programs stored in LCSC

pi: viewing probability for the *i*-th program

$Bp1$: blocking probability for viewing a program stored in LSC

$Bp2$: blocking probability for viewing a program stored in LCSC

$Bp3$: blocking probability for viewing a program stored in CSC

P: per-user request probability

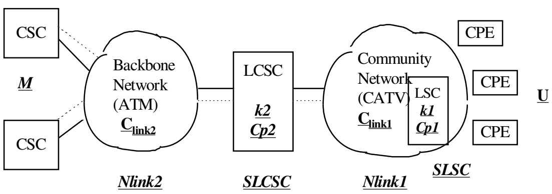

Figure 3 Cost Parameters in the system

Minimum Cost Model

Our goal is to define a cost function to calculate the cost with respect to the number of programs stored in LSC, *k1*, and the number of programs stored in LCSC, *k2*. In the system, the resources required to provide a VOD customer connected to a LSC are dependent on the storage capacity in LSC and LCSC, and the numbers of dedicated links between LSC and CSC and between LCSC and CSC. The storage cost of CSC is neglected because we add it to the cost of links on the backbone network, as the cost of delivering a program on the backbone network is very large than delivery a program on community network. We describe the cost function ,***C,*** as follows:

$$\begin{aligned} \boldsymbol{C} &= \text{storage cost} + \text{network cost} \\ &= (\text{cost of LSC} + \text{cost of LCSC}) \\ &\quad + (\text{cost of community network} + \text{cost of backbone network}) \end{aligned}$$

$$C = Cp1*SLSC + Cp2*SLCSC + C_{link1}*Nlink1 + C_{link2}*Nlink2 \qquad (1)$$

In this paper we assume, for simplicity, R_{stor} is the ratio of Cp1 and Cp2, and R_{link} is the ratio of C_{link1} and C_{link2} as follows:

$$Cp1 = R_{stor}*Cp2 \quad where\ R_{stor} \geq 1$$

$$C_{link1} = R_{link}*C_{link2} \quad where\ \ 0 < R_{link} \leq 1$$

Our cost function ,C, becomes:

$$C= R_{stor}*Cp2*SLSC + Cp2*SLCSC + R_{link} * C_{link2}*Nlink1 + C_{link2}*Nlink2 \qquad (2)$$

In order to overcome the difficulties in keeping up with the continuous variability of the storage and transport costs, it is preferable to make our model independent on the absolute cost values of *Cp1*, *Cp2*, C_{link1} and C_{link2}, but dependent only on their ratio, $C_{link2}/Cp2$. This general approach permits us to take into account any present or future transmission technology used in the transport network and storage device used in the system. If we divide the equation (2) by *Cp2*, we obtain a new cost function ***C′*** as

$$C' = \frac{C}{Cp2} = (R_{stor}*SLSC + SLCSC) + \frac{Cl2}{Cp2}(R_{link} *Nlink1 + Nlink2). \qquad (3)$$

Our goal is to minimize the function ,***C′,*** with respect to the number of programs stored in LSC, *k1*, and the number of program stored in LCSC, *k2*, for the given ratio of $C_{link2}/Cp2$, R_{stor}, and R_{link}.

In our cost model, every program stored in the system is associated with a viewing probability. Suppose that all the programs are sorted in a decreasing order according to their viewing probabilities. Now we determine *SLSC*, *SLCSC*, *Nlink1* and *Nlink2* as follows:

a. Derivation of SLSC and SLCSC

For a single program stored in LSC and associated with a viewing probability *pi*, the number of users requesting the program *i* is given by

$$U * P * pi \qquad (4)$$

Considering a fixed blocking probability for a single program, *Bp1*, we can find that the number of simultaneous sessions to be guaranteed for the program *i* is

$$(1\text{-}Bp1)\cdot U\cdot P\cdot pi$$

Since a single copy of a program stored on LSC guarantees a number of sessions, *Vhd1*, at the same time. The number of program copies needed to satisfy the number of users in (4) is

$$\left\lceil \frac{(1-Bp1)\cdot U\cdot P\cdot pi}{Vhd1} \right\rceil.$$

For the first *k1* video programs stored in the LSC, we can find out the number of program copies in LSC, *SLSC,* as

$$SLSC \;=\; \sum_{i=1}^{k1} \left\lceil \frac{(1-Bp1)\cdot U\cdot P\cdot pi}{Vhd1} \right\rceil$$

And we can also use the previous method to obtain the number of program copies in LCSC, *SLCSC,* as

$$SLCSC \;=\; \sum_{i=k1+1}^{k1+k2} \left\lceil \frac{(1-Bp2)\cdot U\cdot P\cdot pi}{Vhd2} \right\rceil$$

b. Derivation of Nlink1 and Nlink2

The traffic to transport the programs resident in CSC is routed on dedicated links between CSC and LSC. Therefore, the programs stored in LCSC require dedicated links between CSC and LCSC, and also dedicated links between LCSC and LSC. The programs stored in LCSC need only dedicated links between LSC and LCSC. We can find out the number of dedicate links between LSC and LCSC for program *i*, stored in LCSC, is

$$U*P*pi.$$

The number of dedicated links between LSC and LCSC for all program stored in LCSC is

$$U\cdot P\cdot \sum_{i=k1+k2+1}^{M} pi\,.$$

Considering a fixed blocking probability, Bp2, for all programs stored in LCSC, we can find out the number of dedicated links between LCSC and LSC for all programs stored in LCSC as

$$\left\lceil (1-Bp2)\cdot U\cdot P\cdot \sum_{i=k1+1}^{k1+k2} pi) \right\rceil$$

Because *Nlink1* is the number of links between LSC and LCSC, the programs stored in CSC also need the links on the community network. So *Nlink2* have to be added to *Nlink1*, we can obtain *Nlink1* as follows:

$$Nlink1 \;=\; \left\lceil (1-Bp2)\cdot U\cdot P\cdot \sum_{i=k1+1}^{k1+k2} pi) \right\rceil + Nlink2$$

We can also use previous method to obtain the number of links between CSC and LCSC as follows:

$$Nlink2 \;=\; \left\lceil (1-Bp3)\cdot U\cdot P\cdot \sum_{i=k1+k2+1}^{M} pi) \right\rceil$$

After the expressions of *SLSC*, *SLCSC*, *Nlink1* and *Nlink2* are determined, we can look for the value of *k1* and *k2* to minimizes the cost function ***C'***.

3. Results and Implications
Parameter Set

We have to give values for the parameters defined in our minimum cost model in order to calculate the minimum value of the derived cost function. The cost function has been programmed to compute the storage capacity required in the LSC and LCSC and the number of links between service centers. We give the following parameter values:

M: total number of programs in the network(system) = 10000

U: number of customers connected to a LSC = 2000

Vhd1: a single copy of program stored on LSC and LCSC guarantees 2 session at the same time

Vhd2: a single copy of program stored on LSC and LCSC guarantees 1 session at the same time

Bp1: blocking probability to access a program stored in LSC = 0.005

Bp2: blocking probability to access a program stored in LCSC = 0.005

Bp3: blocking probability to access a program stored in CSC = 0.00001

P: per-user request probability = 0.1

The program viewing probability distribution function is described in previous section, with the ratio, α, as 0.98.

Optimal Program Allocation

If the parameter values of *Cp1/Cp2*, C_{link1}/C_{link2}, $C_{link2}/Cp2$ and α have been defined, the cost function ,***C'*** will become a linear function of the number of programs to be stored in the LSC, *k1*, and the number of programs to be stored in the LCSC, *k2*. We assume that *Cp1/Cp2= 3*, C_{link1}/C_{link2}= 0.15, $C_{link2}/Cp2$= 16 and α =0.98 in order to calculate the minimal cost and its corresponding *k1* and *k2*. In figure 6, the 3-D cost-program relationship graph, we can find out the minimum cost, 620, and the corresponding values of *k1*, 75, and *k2*, 133. The corresponding *k1* and *k2* values of the minimum cost give us the programs to be stored inside the LSC and LCSC, and the number of links between service centers.

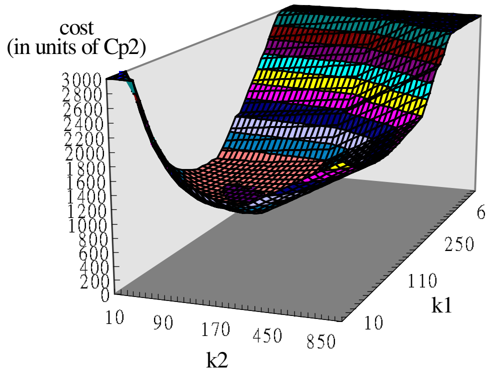

Figure 4 Relationship between *k1,k2* and cost. optimal values: (*k1,k2*,cost)=(75,133,620)

The number of links needed between LSC and LCSC decreases as *k1* increases, and the number of links between LCSC and CSC decreases as *k1+k2* increases. The value of *Nlink1* decreases rapidly as programs with high viewing probabilities are stored in LSC, and the value of *Nlink2* decreases rapidly as program with high viewing probabilities are stored in LCSC. If we store programs with low viewing probabilities in LSC or LCSC, it has only a little impact on *Nlink1* and *Nlink2*.

Influence of Program Viewing Probability

According to the program viewing probability distribution function used in our cost model, the program viewing probability factor, α, plays an important role in our cost model. Figure 5 shows the relationship between minimum cost and α. Figure 6 illustrates the values of *k1* and *k2* that optimize the cost function for the given α. We summarize the results below:

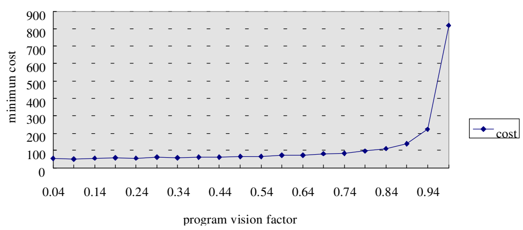

Figure 5 Minimum cost vs. program viewing factor α.

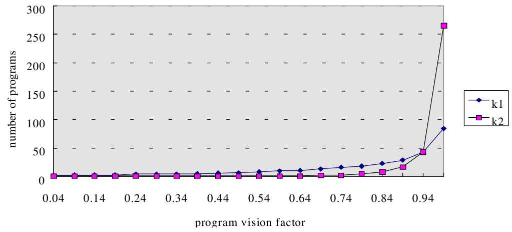

Figure 6 k1 and k2 vs. program viewing factor α

1. The request distribution becomes highly skewed for small α values(close to 0): The requests of users concentrate on a few hot programs. This situation is ideal for VOD service providers, because the VOD service providers only have to store a few high viewing programs inside LSC and LCSC.

2. The request distribution becomes quite uniform for large α values(close to 1): The requests of users do not concentrate on a few programs, but the requests of users are distributed among all programs equally. This is the worst situation for VOD service providers because the numbers of programs stored in LSC and LCSC will be increased in order to satisfy the user requests.

In other words, the cost of a VOD system with a smaller α value will be lower than the cost of a VOD system with a larger α value.

Cost Parameter Adjustment

We assumed, for simplicity, the parameter values of *Cp1/Cp2* and C_{link1}/C_{link2} are equal to R_{stor} and R_{link}. And we also give the value R_{ls2} to $C_{link2}/Cp2$ now. The ratio of *Cp1, Cp2,* C_{link1} and C_{link2} is as follows:

$$Cp1 : Cp2 : C_{link1} : Cp2 = R_{stor} : 1 : R_{link} * R_{ls2} : R_{ls2} \quad (5)$$

We can obtain the ratio of different components used in the VOD system. In this section, we discuss the influences of different R_{stor}, R_{link} and R_{ls2} values on the VOD system.

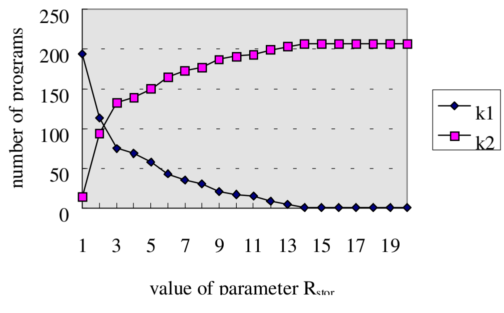

Figure 7 *k1* and *k2* vs. parameter R_{stor}

The parameter R_{stor} is the ratio of the cost of a program copy stored in LSC and the cost of a program copy stored in LCSC. We assume that the cost of a program copy stored in LSC is more than the cost of a program copy stored in LCSC. That is, $R_{stor} \geq 1$. Figure 7 illustrates *k1* and *k2* for the given values of R_{stor}. If we increase the parameter R_{stor}, *k1* will be decreased and *k2* will be increased. But the sum of *k1* and *k2* will be kept at a fixed value for all value of R_{stor}. Because the programs to be removed from in LSC will be moved to LCSC, so *k1+k2* is kept at a fixed value. But the programs in LCSC need extra links on the community network, therefore, *Nlink1* will be increased, and the cost will be increased.

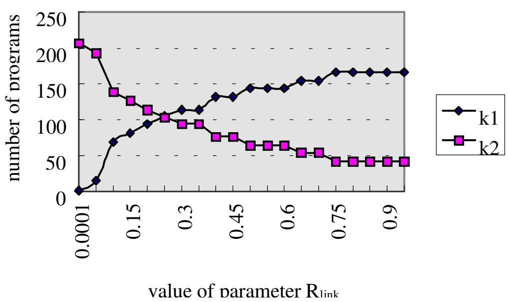

Figure 8 *k1* and *k2* vs. parameter R_{link}

The parameter R_{link} is the ratio of the cost of a program to be carried on the community network and the cost of a program to be carried on the backbone network. In our assumption, the link cost on the backbone network is more than the link cost on the community network. So the range of R_{link} is between zero and one. Figure 8 illustrates the relationship between *k1*, *k2* and R_{link}. As R_{link} approaches 1, the link cost on the community network is almost equal to the link cost on the backbone network. So *k1* will be increased but *k2* will be decreased. Because the programs to be removed from the LCSC will be moved into LSC in order to avoid the cost of links on the community network. So k1+k2 is kept at a fixed value for variable R_{link}. But the storage cost in LSC is larger than the storage cost in LCSC, therefore, the cost of system will be increased.

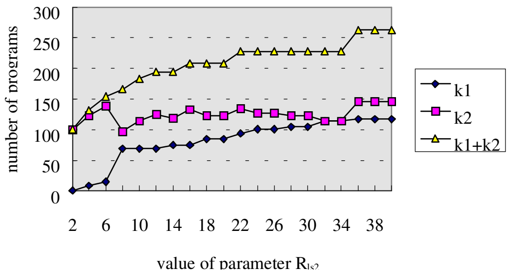

Figure 9 *k1*, *k2* and *k1+k2* vs. parameter R_{ls2}

The parameter R_{ls2} is the ratio of the cost of a program to be carried on the backbone network and the cost of a program copy to be stored in LCSC. Figure 9 illustrates the relationship between *k1, k2, k1+k2* and R_{ls2}. In our assumption, the value of parameter R_{ls2} is greater than 1. Clearly, if R_{ls2} increases, the cost of programs carried on the backbone and community network will be increased. The *k1, k1+k2* and cost will be increased, but *k2* does not have an obvious change. Because the transmission cost of community network increases, the number of programs to be stored in LCSC will be decreased in order to avoid link cost. And at the same time, the transmission cost of backbone network increases, some programs will be moved from CSC to LCSC in order to avoid the link cost on the backbone network.

4. Program Re-allocation Algorithm

From the previous results, the program viewing probability distribution function has a strong influence on the program allocation of the system. Different distributions of users' interest result in different combinations of *k1* and *k2* values. In other words, the storage capacity of service centers(LSC and LCSC) will be changed according to users' request pattern. If we need to keep the values of *k1* and *k2* optimizing the cost function, the capacities of LSC and LCSC will be changed at every re-allocation period. But there may be only a little variation between two re-allocation periods. Under this assumption, there are two kinds of service centers to be used in the system. We describe them as follows:

LSC and LCSC with Variable Capacities

In this situation, the system will maintain the system with a minimum cost. But the capacity of

LSC and LCSC will be changed in every program re-allocation period. The algorithm performs from step1 to step8.

Algorithm 1

Program re-allocation in **LSC:**

step1(Start): Find out the viewing probabilities of all programs in the system according to the current program viewing probability distribution function, and find out *k1* and *k2* that optimize the cost function.

step2(Remove): If the program is stored in LSC before **step1**, but its viewing probability is less than the probability of the *k1* -th program after **step1**, the program will be uploaded to LCSC and the capacity used by the program will be released.

step3(Adjust): If the viewing probability of the program is still more than the viewing probability of the *k1*-th program after **step1**, we adjust the number of program copies according to its viewing probability.

step4(Add): If the program is stored in LCSC, but its viewing probability is more than the probability of *k1*-th program after **step1**, it will be downloaded from LCSC and we duplicate the program copies , if necessary, according to its viewing probability.

Program re-allocation in **LCSC:**

step5(Remove): If the program is stored in LCSC before **step1**, but its viewing probability is not between the viewing probability of the first program and the k1-th program after **step1**, the capacity used by the program will be released.

step6(Adjust): If the program to be stored in LCSC before **step1**, and its viewing probability is still between the probability of the (*k1*+1)-th and the (*k1*+*k2*)-th program after **step1**, the capacity used by the program will be adjusted.

step7(Add, from CSC): If the viewing probability of the program is less than the (*k1*+*k2*)-th program's before **step1**, but its viewing probability is between the viewing probability of the *k1*-th and the (*k1*+*k2*+1)-th program's after **step1**,the program will be downloaded from CSC, and we will duplicate the number of program copies according to it's viewing probability.

step8(Add, from LSC): If the viewing probability of the program is more than the (*k1*+*k2*)-th program's before **step1**, but its viewing probability is between the viewing probability of the *k1*-th and the (*k1*+*k2*+1)-th program's after **step1**, the program will be uploaded from LSC, and we will duplicate the number of program copies according to it's viewing probability.

LSC and LCSC with Fixed Capacities

The system has fixed capacities, but the system is not operating at a minimum cost. We use two counters(*counter1* and *counter2*) to record the capacities of LSC and LCSC, and an index *i* to indicate the program index. Let *counter1* and *counter2* equal to 0, and index *I* equal to 1 at the beginning. We also set the fixed capacities of LSC and LCSC as *C_LSC* and *C_LCSC,* respectively. The algorithm here is similar to the previous algorithm.

Algorithm 2

(1) Find out the viewing probabilities of all program in our system according to the program viewing probability distribution function. And calculate the numbers of program copies of all programs using the cost function.

Program re-allocation in **LSC:**

(2) ***if*** the capacity of program *i* + *counter1* ≤ *C_LSC,*

then 1. execute **step2, step3** or **step4** of algorithm 1,

2. *counter1* = *counter1* + capacity of program *i* ,

3. *i* = *i* + 1, and repeat **(2),**

else go to **(3).**

Program re-allocation in **LCSC:**

(3) ***if*** the capacity of program *i* + *counter2* ≤ *C_LCSC,*

then 1. execute **step5, step6, step7** or **step8** of algorithm 1,

2. *counter2* = *counter2* + capacity of program *i* ,

3. *i* = *i* + 1, and repeat **(3),**

else re-allocation is completed.

5. Conclusion and Future Work

In this paper we have proposed a three-level hierarchical network storage architecture for delivering the VOD service. One of the challenges in designing a cost-effective VOD service system that supports thousands of video streams is allocating the programs optimally among various service centers. An under-utilized system is costly, but an overloaded system causes service interruption or performance degradation. An optimal design requires the appropriate allocation of programs to various service centers based on the customer request probabilities. We use a minimum cost method to find out the storage capacities in LSC and LCSC and the numbers of links between LSC and LCSC, and between LCSC and CSC. Under the condition of our given parameter values, our cost function becomes a linear function of number of programs to be stored in LSC and number of programs to be stored in LCSC. We can obtain the minimum cost of the system and the optimal combination of numbers of programs to be stored in LSC and LCSC.

According to the results of our analysis, the program viewing probability distribution function has a strong influence on the resource allocation of the system. $k1$(The number of programs to be stored in LSC) increases as R_{link}(the ratio of the cost of a program to be delivered on community network and on backbone network) or R_{ls2}(the ratio of the cost of a program to be delivered on backbone network and the cost of a program copy to be stored in LCSC) increases, but $k1$ decreases as R_{stor}(the ratio of the cost of a program copy to be stored in LSC and LCSC) increases. $k2$(The number of programs to be stored in LCSC) increases as R_{stor} increases, but $k2$ increases as R_{link} or R_{ls2} decreases. $k1+k2$(sum of the numbers of programs to be stored in LSC and LCSC) will be kept at a fixed value for all values of R_{stor} or R_{link}, but it increases as R_{ls2} increases. The cost of system increases as R_{stor}, R_{link} or R_{ls2} increases.

The program viewing probability distribution function also plays an important role in program allocation. We need a program viewing probability model which is closer to the practical distribution of user requests. This is an important issue in our future work. Another problem is the capacity of LSC and LCSC. In our minimum cost model, the numbers of programs to be stored in LSC and LCSC change as the request pattern of users changes. So we obtain various combinations of the numbers of programs to be stored in LSC and LCSC every time. The capacities of LSC and LCSC will be changed at every re-allocation time. Though, we have proposed two program reallocation algorithms for the system, the implementation and analysis of these algorithms require further studies.

References

[1]A. D. Gelman, H. Kobrinski, L. S. Smoot, S. B. Weinstein, “A store-and-forward architecture for video-on-demand services”, Proc. IEEE ICC, 1991, pp. 27.3.1-27.3.5.

[2]John Woodward, Tom Helmes, Harry Mussman Steve Walker, Len Ulbricht and Dean Casey, “ A broadband network archecture for residential video service featuring distributed control of broadband seitches and video dial tone functionality”, Proc. IEEE ICC, 1993, pp. 853-857.

[3]Daniel Deloddere, Willem Verbiest, and Henri Verhille, “ Interactive Video On Demand ”, IEEE communications magazine, May 1994, pp. 82-88.

[4]W.D. Sincoskie , “ System architecture for a large scale video on demand service”, Computer Networks and ISDN Systems, 1991, pp. 155-162.

[5]D.C. Thomas ,“Prospects for Interactive Video-on-Demand”, IEEE Multimedia, Fall 1994, pp. 14-24.

[6]L. De Giovanni, A. M. Langellotti, L. M. Patitucci, L. Petrini, “ Dimensioning of Hierarchical Storage for Video on Demand Services ”, Proc. IEEE ICC, 1994, pp. 1739-1743.

[7]Yurdaer N. Doganata and Asser N. Tantawi, “ Making a Cost - Effective Video Server”, IEEE Multimedia, Winter 1994, pp. 22-30.

[8]T.D.C. Little, D.Venkatesh, “Popularity-based assignment of movies to storage devices in a video-on-demand system”, Multimedia Systems, Springer-Verlag 1995, pp. 280-287.

[9]Milind M.Buddhikot, Guru M.Parulkar, Jerome R.Cox.Jr, “Design of a large scale multimedia storage server”, Computer Networks and ISDN System, 27, 1994, pp. 503-517.

[10]D. Patterson etal., “ A case for Redundant Array of Inexpensive Disk(RAID) ” Proc. ACM Conf. on Management of Data (SIGMOD), 1988, pp. 109-116.