A New Fixed-Delay Broadcasting Protocol for Near Video-on-Demand Services

YU-WEI CHEN
Department of Computer and Information Science
Aletheia University
No. 32, Chen-Li Street, Tamsui, Taipei, 25103 Taiwan
REPUBLIC OF CHINA
http://www.cis.au.edu.tw

Abstract: - This work presents a novel *fixed-delay* broadcasting protocol for near video-on-demand service. Consider a film S is divided into n equal parts $\{S_1, S_2, \ldots, S_n\}$ and played through k channels. Each segment S_i must appear at least once every $i+c-1$ segments rather than i segments. The value of parameter c means that the users have to wait the time of c segments. Each channel of k channels is first partitioned into subchannels by a heuristic strategy. Then, a greedy approach is applied to assign the n segments to k channels. The waiting time is c times the duration of one segment. As a result, the proposed method outperforms the previous broadcasting methods in terms of the maximum waiting time, for the same number of channels.

Key-Words: - Broadcasting, Fixed-delay, Multimedia, Near Video-on-Demand, Protocol, Waiting Time.

1 Introduction

The so-called Video-on-Demand (VOD) system refers to the use of a combination of a television set and a set-top-box or computer, lined via the Internet to a film provider. The system allows the user to choose which film to watch and utilize video-cassette-recorder capabilities such as playback, forward, rewind, pause and others. One of the simplest ways to utilize the various video-cassette-recorder capabilities is for the provider to set up a dedicated channel for the user when a request to view a film is made. This VOD system is called the true VOD system

The main problem of the True VOD system lies in the requirement for an extremely large bandwidth for transmitting a large amount of information. Many researchers have proposed various solutions to reduce the need for bandwidth. They can be broadly categorized into three types for VOD systems – batching [1]-[3], stream tapping [4]-[6] or patching [7] and broadcasting [8]-[21].

Of the three VOD types, broadcasting saves much more bandwidth than the other two. The broadcasting approach works by first dividing the film into several segments, and then allowing the server to play them repeatedly in a few specified channels. Should a user wish to view the film, the greatest waiting time will be the duration of one such segment. Different methods [12]-[18] are used to divide the film into different numbers of segments, so waiting times differs. The user must also have enough buffer space to store the downloaded segments for continuous playback. This system is also known as the near VOD system.

Recently, Pâris et al. [22] and Chen [23] proposed "fixed-delay" broadcasting protocols. Should a user wish to view a film, the greatest waiting time will be the duration of m segments rather than one segment. The two methods [22], [23] have shorter maximum waiting time than those in [12][15]-[18]. In addition, Pâris et al. [24] proposed another fixed-delay broadcasting protocol by using previews to reduce the cost of VOD services.

This work presents a novel *fixed-delay* broadcasting protocol for near VOD service. Consider a film S is divided into n equal parts $\{S_1, S_2, \ldots, S_n\}$ and played through k channels. Each segment S_i must appear at least once every $i+c-1$ segments rather than i segments. Each channel of k channels is first partitioned into subchannels by a heuristic strategy. Then, a greedy approach is applied to assign the n segments into k channels. The user simultaneously downloads the film segments from k channels into the set-top-box (STB) and plays them in order. All users wait for a fixed delay T before watching the video. This waiting time T is c times the duration of one segment. As a result, the proposed method outperforms the previous broadcasting methods [22], [23] in terms of the maximum waiting time, for the same number of channels.

This paper is organized as follows. In Section 2, preliminaries are broadly introduced. The new broadcasting protocol is proposed and analyzed in

Section 3. Finally, in Section 4, the conclusions are addressed.

2 Preliminaries

2.1 Fixed-Length Segment-Scheduling Problem

Fast Broadcasting [12][15], Pagoda broadcasting [16][17], and Recursive frequency splitting broadcasting [18] are all problems of Fixed-length Segment-Scheduling. In such a problem, the film S is always divided into n equal parts (S_1, S_2, …, S_n) and played back through k channels, at the same time assuming each channel's bandwidth is just sufficient to fit the usage rate of the film during normal playback and that the user can receive information from k channels. Since before watching film segment S_i the first i-1 segments S_1, S_2, …, S_{i-2}, and S_{i-1} have to be watched, segment S_i has to at least appear once in the time every i segments is played to ensure proper playback without breaks in between. Therefore segment S_i at least needs to use up $1/i$ of channel bandwidth. Under the condition of uninterrupted playback, the largest number of segments that can be divided is n, and n satisfies:

$$\frac{1}{1}+\frac{1}{2}+\ldots+\frac{1}{n}=k<\frac{1}{1}+\frac{1}{2}+\ldots+\frac{1}{n+1}.$$

2.2 Traditional Broadcasting Protocols

2.2.1 Stagger Broadcasting

Assume we want to play a film S of time length L through k channels {C_0, C_1, …, C_{k-1}}. In the initial time, stagger broadcasting protocol [9] let the whole film will repeatedly transmit on C_0. After the time $i \cdot L/k$, $1 \le i \le k$-1, the same whole film will also be transmitted on C_i periodically. The maximum waiting time is L/k.

2.2.2 Fast Broadcasting

The basic concept of fast broadcasting method [12] is described as follows. A film S is first partitioned into 2^k-1 segments {S_1, S_2, …, $S_{2^{k-2}}$, S_{2^k-1}}. Segment S_1 will repeatedly transmit on C_0. The 2^i segments {S_{2^i}, S_{2^i+1}, …, $S_{2^{i+1}-1}$} repeatedly transmit in order on C_i for $1 \le i \le k$-1.

2.2.3 Pagoda Broadcasting

The basic concept of pagoda broadcasting method [17] is described as follows. A film S is first partitioned into n segments {S_1, S_2, …, S_n} where $n = 4(5^{(k/2)-1})$-1 if k is even and $n = 2(5^{\lfloor k/2 \rfloor})$ if k is odd. Segment S_1 will repeatedly transmit on C_0. Let q=2r-1 for $r \ge 1$ and the index of S_z is smaller than those of the other segments broadcasted on C_q. The z/2 segments {S_z, S_{z+1}, …, $S_{3z/2-1}$} repeatedly transmit in order on the odd slots on C_q. The z segments {S_{2z}, S_{2z+2}, S_{2z+4}, …, S_{3z-2}, S_{2z+1}, S_{2z+3}, S_{2z+5}, …, S_{3z-1}} repeatedly transmit in order on the even slots on C_q.

After transmitting segments on C_q, the following introduces the segments scheduling on C_{q+1}. The z/2 segments {$S_{3z/2}$, $S_{3z/2+1}$, …, S_{2z-1}} repeatedly transmit in order on the slots $3i$+1, $0 \le i$, on C_{q+1}. The z segments {S_{3z}, S_{3z+2}, S_{3z+4}, …, S_{4z-4}, S_{4z-2}, S_{3z+1}, S_{3z+3}, …, S_{4z-3}, S_{4z-1}} repeatedly transmit in order on the slots $3i$+2 on C_{q+1}. The z segments {S_{4z}, S_{4z+2}, S_{4z+4}, …, S_{5z-4}, S_{5z-2}, S_{4z+1}, S_{4z+3}, …, S_{5z-3}, S_{5z-1}} repeatedly transmit in order on the slots $3i$+3 on C_{q+1}.

If k is odd, the pagoda method is finished. Otherwise, if k is even, the z segments {S_z, S_{z+1}, S_{z+2}, …, S_{2z-1}} repeatedly transmit in order on C_{k-1}.

Fig. 1 and Fig. 2 show the Pagoda scheme's scheduling for three and four channels, respectively.

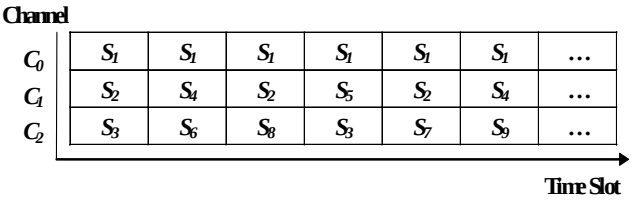

Fig. 1. Pagoda scheme's scheduling for three channels.

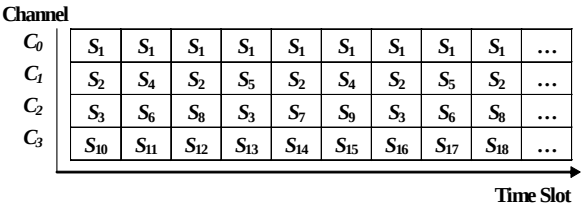

Fig. 2. Pagoda scheme's scheduling for four channels.

2.2.4 Recursive Frequency Splitting Broadcasting

The basic concept of recursive frequency splitting (RFS) broadcasting scheme [18] is described as follows. Tseng, Yang, and Chang first define the slot sequence SS(C_i, p, q) as an infinite sequence of time slots [p, p+q, p+2q, …] belonging to channel C_i, beginning at slot p, and repeating infinitely with a period of q slots, where C_i is one of the k channels, $p \ge 0$ is an integer, and $q \ge 1$ is an integer, $0 \le p \le q - 1$.

Initially, let POOL = {SS(C_0, 0, 1), SS(C_1, 0, 1), SS(C_2, 0, 1), …, SS(C_{k-1}, 0, 1)} denote the set of free channels and let j be the index of segment. The initial value of j is 1. Second, pick a slot sequence SS(C_i, p,

q) with the smallest value of j mod q from POOL such that $q \le j$. Let POOL = POOL – {SS(C_i, p, q)}. Third, split SS(C_i, p, q) into {SS(C_i, p, αq), SS(C_i, p+q, αq), SS(C_i, p+2q, αq), …SS(C_i, p+(α-1)q, αq)} where $\alpha = \lfloor j/q \rfloor$. Segment S_j is broadcasted on the slots in SS(C_i, p, αq). Do the union POOL = POOL $\cup$ { SS(C_i, p+xq, αq) | $1 \le x \le \alpha$-1}. If POOL is not empty, then increase j by one and go to the second phase. Otherwise, terminate this process and output the value of j. Fig. 3 illustrates the result of RFS algorithm with four channels.

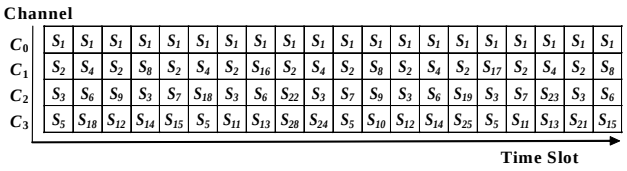

Fig. 3. Recursive frequency-splitting scheme's scheduling for four channels.

2.3 Fixed-Delay Broadcasting Protocols

2.3.1 Fixed-Delay Pagoda Broadcasting

Pâris et al. [22] proposed a fixed-delay pagoda broadcasting (FDPB) protocol. With the FDBP protocol, segments S_i must be transmitted at least once per m+i-1 slots, where m is an integer $m \ge 1$. Consider a case in which m=9; the basic concept of FDPB protocol is thus described as follows. Segment S_1 is the first segment to be broadcast on channel C_0 and will need to be transmitted at least once every nine slots. The first channel C_0 is partitioned into three ($=\sqrt{9}$) subchannels. The three segments S_1, S_2 and S_3 will be assigned to the first subchannel on C_0 and each is repeated once per nine slots. The four segments S_4 to S_7 are assigned to the second subchannel on C_0 and each is repeated once per 12 slots. The five segments S_8 to S_{12} are assigned to the third subchannel on C_0 and each is repeated once per 15 slots.

Segment S_{13} is the first segment to be broadcast on channel C_1 and must be transmitted at least once per 21 (=13+9-1) slots. The number 21 is not a square and the closest square is $25=5^2$, so channel C_1 is partitioned into five subchannels. Segments S_{13} to S_{16} are assigned to the first subchannel on C_1. Similarly, the four groups, Segments S_{17} to S_{21}, Segments S_{22} to S_{27}, Segments S_{28} to S_{34}, and Segments S_{35} to S_{42}, are assigned to the other four subchannels on C_1 while each of the segments of the four groups will be repeated once every 25, 30, 35 and 40 slots, respectively.

Segment S_{43} is the first segment to be broadcast on channel C_2 and must be transmitted at least once per 51 (=43+9-1) slots. Therefore, channel C_2 is partitioned into seven subchannels and Segments S_{43} to S_{116} are assigned to C_2. The FDPB protocol can achieve segment-to-channel mapping for any k channels when the above concept is applied repeatedly. Table 1 illustrates the result of FDPB algorithm with seven channels for m=9.

TABLE 1
Fixed-Delay Pagoda Broadcasting Mapping for m=9

Channel	Number of Subchannels	First Segment	Last Segment
C_0	3	S_1	S_{12}
C_1	5	S_{13}	S_{42}
C_2	7	S_{43}	S_{116}
C_3	11	S_{117}	S_{308}
C_4	18	S_{309}	S_{814}
C_5	29	S_{815}	S_{2168}
C_6	47	S_{2169}	S_{5810}

2.3.2 Enhanced Recursive Frequency Splitting Broadcasting

The basic concept of enhanced recursive frequency splitting (ERFS) broadcasting scheme [23], called c-ERFS, is described as follows. The number of available channels is k. Segment S_1 must appear once per c segments, rather than per one segment. The c-ERFS uses the definition of the slot sequence SS(C_i, p, q) in [18].

Initially, let POOL = {SS(C_i, 0, 1) | $0 \le i \le k$-1} denote the set of free channels. Each channel is partitioned into $\lfloor \sqrt{c} \rfloor$ subchannels. Thus, POOL = {SS(C_i, δ, $\lfloor \sqrt{c} \rfloor$) | $0 \le i \le k$-1, $0 \le \delta \le \lfloor \sqrt{c} \rfloor$-1}. The initial value of j is c for $c \ge 1$. Second, pick a slot sequence SS(C_i, p, q) with the smallest value of j mod q from POOL such that $q \le j$. Let POOL = POOL – {SS(C_i, p, q)}. Third, split SS(C_i, p, q) into {SS(C_i, p+xq, αq) | $0 \le x \le \alpha$-1, $\alpha = \lfloor j/q \rfloor$}. Segment S_{j-c+1} is broadcasted on the slots in SS(C_i, p, αq). Set POOL = POOL $\cup$ { SS(C_i, p+xq, αq) | $1 \le x \le \alpha$-1}. If POOL is not empty, then increase j by one and go to second phase. Otherwise, terminate this process and output the value of j-c+1. Fig. 4 shows the example schedules of segments obtained using 2-ERFS with three channels.

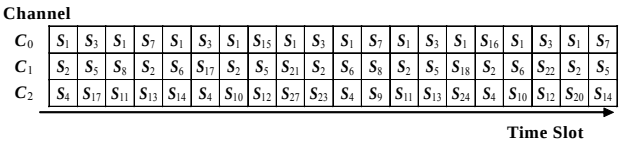

Fig. 4. Schedule of segments for 2-ERFS with three channels.

3 Proposed Broadcasting Protocol

The basic concept of the proposed fixed-delay recursive-frequency-splitting broadcasting (FDRFS) protocol is described as follows. Each S_j does not need to appear at least once per j segments; rather, each S_j must appear at least once per $j+c-1$ segments for j, $c\geq 1$. Before scheduling the segments on available channels, each channel is first divided into different subchannels. Letting $x=c$, channel C_0 is divided into $\lfloor\sqrt{x}\rfloor$ subchannels. Letting $x= x+\lfloor\sqrt{x}\rfloor$, channel C_1 is divided into $\lfloor\sqrt{x}\rfloor$ subchannels. Recursively, each channel is divided into different subchannels.

Then, the same concept as in [23] is improved and applied to schedule the segments, increasing the number of cut video segments. The waiting time for the user decreases accordingly. The difference between this paper and [23] is the *channel-splitting* strategy. For example, a slot sequence SS(C_i, p, q) with the smallest value of j mod q is selected for S_{j-c+1}, $q \leq j$. The prime factorization of j/q is $a_1*a_2*\ldots*a_m$ for $a_x \leq a_y$, $1\leq x\leq y\leq m$. The slot sequence SS(C_i, p, q) is first split into {SS(C_i, $p+xq$, a_1q) | $0\leq x\leq a_1-1$}. Then, SS(C_i, $p+(a_1-1)q$, a_1q) is split into {SS(C_i, $p+x(a_1-1)q$, a_1a_2q) | $0\leq x\leq a_2-1$}. Further, SS(C_i, $p+(a_1-1)(a_2-1)q$, a_1a_2q) is split into {SS(C_i, $p+x(a_1-1)(a_2-1)q$, $a_1a_2a_3q$) | $0\leq x\leq a_3-1$}. Recursively do the splitting processing, finally, SS(C_i, $p+(a_1-1)(a_2-1)\ldots(a_{m-1}-1)q$, $a_1a_2\ldots a_{m-1}q$) is split into {SS(C_i, $p+x(a_1-1)(a_2-1)\ldots(a_{m-1}-1)q$, $a_1a_2a_3\ldots a_mq$) | $0\leq x\leq a_m-1$}. Consequently, SS(C_i, p, q) can be split into

$$\bigcup_{z=1}^{m}\bigcup_{x=0}^{a_z-2}\left\{SS(C_i, p+xq\cdot_{y=1}^{z-1}(a_y-1), q\cdot_{y=1}^{z}a_y)\right\}$$

$$\bigcup SS(C_i, p+q\cdot_{y=1}^{m}(a_y-1), q\cdot_{y=1}^{m}a_y),$$

where $\cdot_{y=1}^{z-1}(a_y-1)=1$ if z=1.

The proposed method can be implemented using the following steps.

Inputs: positive integers k and c. The number of available channels is k. Segment S_j must appear once per $j+c-1$ segments.

Outputs: the largest number of segments that can be divided.

Algorithm c-FDRFS

Step 1. Let POOL = {SS(C_i, 0, 1) | $0 \leq i \leq k-1$} denote the set of free channels. Let $x=c$ and $i=0$.

Step 2. Channel C_i is partitioned into $\lfloor\sqrt{x}\rfloor$ subchannels {SS(C_i, δ, $\lfloor\sqrt{x}\rfloor$) | $0\leq\delta\leq\lfloor\sqrt{x}\rfloor-1$}. Let $x= x+\lfloor\sqrt{x}\rfloor$ and $i=i+1$. If $i\leq k-1$, then redo Step 2. Otherwise, go to Step 3.

Step 3. Set $j = c$ for $c\geq 1$. Segment S_{j-c+1} must appear at least once per j segments.

Step 4. Select a slot sequence SS(C_i, p, q) with the smallest value of j mod q from POOL such that $q \leq j$. Let POOL = POOL – {SS(C_i, p, q)}.

Step 5. Let the prime factorization of value j/q as $a_1*a_2*\ldots*a_m$. Split SS(C_i, p, q) into

$$\bigcup_{z=1}^{m}\bigcup_{x=0}^{a_z-2}\left\{SS(C_i, p+xq\cdot_{y=1}^{z-1}(a_y-1), q\cdot_{y=1}^{z}a_y)\right\}$$

$$\bigcup SS(C_i, p+q\cdot_{y=1}^{m}(a_y-1), q\cdot_{y=1}^{m}a_y).$$

Segment S_{j-c+1} is broadcasted on the slots in

$$SS(C_i, p+q\cdot_{y=1}^{m}(a_y-1), q\cdot_{y=1}^{m}a_y).$$

Set POOL = POOL $\cup$

$$\bigcup_{z=1}^{m}\bigcup_{x=0}^{a_z-2}\left\{SS(C_i, p+xq\cdot_{y=1}^{z-1}(a_y-1), q\cdot_{y=1}^{z}a_y)\right\}.$$

Step 6. If POOL is not empty, then increase j by one and go to Step 4. Otherwise, terminate this process and output the value of $j-c+1$.

Since segment S_i must appear at least once per $i+c-1$ slots to ensure proper playback without breaks, segment S_i at least needs to use up $1/(i+c-1)$ of channel bandwidth. For uninterrupted playback, the largest number of segments that can be divided is n, and the upper bound on n satisfies:

$$\frac{1}{c}+\frac{1}{c+1}+\ldots+\frac{1}{n}=k<\frac{1}{c}+\frac{1}{c+1}+\ldots+\frac{1}{n+1}.$$

Table 2 compares previous FDPB [22], ERFS [23], the proposed FDRFS protocol and the upper bound in terms of the total segments.

In the proposed protocol, each S_j must appear at least once per $j+c-1$ segments for j, $c\geq 1$. Therefore, the user must wait for the time between $c-1$ and c segments to ensure proper playback without breaks. For a fair comparison, the total number of segments is divided by c, whose division can be thought of as indicating that c segments are packed into a large segment. Table 3 compares previous methods and the proposed method in terms of the total segments. The proposed method clearly outperforms the previous methods in terms of the longest waiting time.

TABLE 2
Total Numbers of Segments for FDPB [22], ERFS [23], Proposed Protocol and Upper Bound

Number of Channels	1	2	3	4	5	6	7
9-FDPB [22]	12	42	116	308	814	2168	5810
100-FDPB [22]	156	565	1650	4563	12418	33684	91321
9-ERFS [23]	12	45	134	383	1055	2778	7789
100-ERFS [23]	148	575	1766	4963	13649	36735	99708
9-FDRFS	12	45	139	390	1113	3048	8350
100-FDRFS	151	574	1778	5039	13922	37794	102608
9-Bound	15	55	163	456	1254	3423	9319
100-Bound	171	636	1900	5334	14668	40042	109016

TABLE 3
Total Numbers of Segments for Previous Methods and Presented Method

Number of Channels	1	2	3	4	5	6	7
Staggered [9]	1	2	3	4	5	6	7
Fast [12]	1	3	7	15	31	63	127
Pagoda [16][17]	1	3	9	19	49	99	249
New Pagoda [16]	1	3	9	26	66	172	422
RFS [18]	1	3	9	25	73	201	565
4-RFS [18]	1	3	9	26	73	201	565
9-FDPB [22]	1.33	4.66	12.88	34.22	90.44	240.88	645.55
100-FDPB [22]	1.56	5.65	16.50	45.63	124.18	336.84	913.21
9-ERFS [23]	1.33	5.00	14.88	42.55	117.22	308.66	865.44
100-ERFS [23]	1.48	5.75	17.66	49.63	136.49	367.35	997.08
9-FDRFS	1.33	5	15.44	43.33	123.67	338.67	927.78
100-FDRFS	1.51	5.74	17.78	50.39	139.22	377.94	1026.08

Like the following methods [12][16]-[18][22][23], the client requires buffer to store portion of the video. The bound of the buffer requirement can be analysed by applying the same analysis concept [25]. Consider a film S is divided into n equal segments and played through k channels. Each segment S_i must appear at least once every $i+c-1$ time slots. Since client receives k segments every time slot but consuming only one segment and segment S_j is not to buffer after watching, the maximum buffer requirement must be bounded by:

$$B_k = \max_{0<i=n}\left\{k \cdot (i+c-1) - \sum_{j=1}^{i}\left\lfloor \frac{i+c-1}{p_{k,j}} \right\rfloor\right\}$$

where $p_{k,j}$ denotes the time period of the slot sequence assigned to the segment S_j when using k channels.

4 Conclusion

This work has presented a fixed-delay recursive-frequency-splitting broadcasting protocol for near video-on-demand service. The proposed method outperforms the previous broadcasting methods in terms of the maximum waiting time, for the same number of channels. The bound of the maximum buffer requirement is analysed too.

References:

[1] K. C. Almeroth and M. H. Ammar, On the Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service, *IEEE Journal on Selected Areas in Communications*, Vol. 14, 1996, pp. 1110-1122.

[2] A. Dan, P. Shahabuddin, D. Sitaram, and D. Towsley, Channel Allocation under Batching and VCR Control in Video-on-Demand Systems, *Journal of Parallel and Distributed Computing*, Vol. 30, 1995, pp. 168-179.

[3] A. Dan, D. Sitaram, and P. Shahabuddin, Dynamic Batching Policies for an on-Demand Video Server, *Multimedia Systems*, Vol. 4, 1996, pp. 112-121.

[4] S. W. Carter and D. D. E. Long, Improving Video-on-Demand Server Efficiency through Stream Tapping," *In Proc. of the Sixth International Conf. on Computer communications and Networks* (*ICCCN '97*), Las Vegas, NV, USA, 1997, pp. 200-207.

[5] S. W. Carter and D. D. E. Long, Improving Bandwidth Efficiency on Video-on-Demand Servers, *Computer Networks*, Vol. 20, 1999, pp. 99-111.

[6] S. W. Carter, D. D. E. Long, and J. -F. Pâris, An Efficient Implementation of Interactive Video-on-Demand, *In Proc. of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, San Francisco, CA, 2000, pp. 172-179.

[7] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," *Proc. 6th ACM Multimedia Conf.*, 1998, pp. 191-200.

[8] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," *In IEEE Proc. of the International Conf. on Multimedia Computing and Systems*, 1996, pp. 118-126.

[9] T. Chiueh and C. Lu, A Periodic Bbroadcasting Approach to Video-on-Ddemand Service, *International Society for Optical Engineering*, Vol. 2615, 1995, pp. 162-169.

[10] L. Gao, J. Kurose, and D. Towsley, Efficient Schemes for Broadcasting Popular Videos, *In International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 1998, pp. 317-329.

[11] K. A. Hua and S. Sheu, Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems," *In ACM SIGCOMM'97*, Vol. 27, 1997, pp. 89-100.

[12] L. -S. Juhn and L. –M. Tseng, Fast Broadcasting for Hot Video Access, *In Real-Time Computing Systems and Applications*, 1997, pp. 237-243.

[13] L. -S. Juhn and L. -M. Tseng, Harmonic Broadcasting for Video-on-Demand Service, *IEEE Trans. on Broadcasting*, Vol. 43, 1997, pp. 268-271.

[14] L. -S. Juhn and L. -M. Tseng, Enhanced Harmonic Data Broadcasting and Receiving Scheme for Popular Video Service, *IEEE Trans. on Consumer Electronics*, Vol. 44, 1998, pp. 343-346.

[15] L. -S. Juhn and L. -M. Tseng, Fast Data Broadcasting and Receiving Scheme for Popular Video Service, *IEEE Trans. on Broadcasting*, Vol. 44, 1998, pp. 100-105.

[16] J. -F. Pâris, A Simple Low-Bandwidth Broadcasting Protocol, *Proc. of the 8th International Conf. on Computer Communications and Networks* (*IC3N'99*), Boston-Natick, MA. 1999, pp. 118-123.

[17] J. -F. Pâris, S. -W. Carter, and D. -D. Long, A Hybrid Broadcasting Protocol for Video on Demand, *In Multimedia Computing and Networking*, 1999, pp. 317-326.

[18] Y. -C. Tseng, M. -H. Yang, and C. -H. Chang, A Recursive Frequency-Splitting Scheme for Broadcasting Hot Videos in VOD Service, *IEEE Trans. on Communications*, Vol. 50, 2002, pp. 1348-1355.

[19] S. Viswanathan and T. Imielinski, Metropolitan Area Video-on-Demand Service Using Pyramid Broadcasting, *IEEE Multimedia Systems*, Vol. 4, 1996, pp. 197-208.

[20] Y. W. Chen, Y. T. Lee, and M. H. Tsai, A Broadcasting Scheme with Supporting VCR Functions for Near Video-on-Demand Systems, *Proc. of the* 9*th International Conf. on Distributed Multimedia Systems*, Miami, Florida, USA, 2003, pp. 737-742.

[21] Y. W. Chen and L. R. Han, A New Broadcasting Scheme with Supporting VCR Functions for Near Video-on-Demand Systems," *Proc. of the* 2004 *International Conf. on Imaging Science, Systems, and Technology*, Las Vegas, Nevada, USA, 2004, pp. 163-169.

[22] J. –F. Pâris, A Fixed-Delay Broadcasting Protocol for Video-on-Demand," *Proc. of the* 10*th International Conf. on Computer Communications and Networks* (*ICCCN'*01), Oct. 2001, pp. 418-423.

[23] Y. W. Chen, An Enhanced Recursive Frequency Splitting Broadcasting Algorithm for Near Video-on-Demand Services, *Information Processing Letters*, Vol. 92, 2004, pp. 299-302.

[24] J. –F. Pâris, Using Previews to Reduce the Cost of Video-on-Demand Services, *J. Research on Computing Science*, Vol. 13, 2005, pp. 179-189.

[25] J. P. Sheu, H. L. Wang, C. H. Chang and Y. C. Tseng, A Fast Video-on-Demand Broadcasting Scheme for Popular Videos, *IEEE Trans. on Broadcasting*, Vol. 50, No. 2, June 2004, pp. 120-125.