# Optimising upload bandwidth for quality of VCR operations in P2P VoD systems

Tzu-Meng Chung, Shih-Chieh Huang, Chung-Ta King* and Chiu-Ping Chang

Department of Computer Science,
National Tsing Hua University,
Kuang-Fu Road,
Hsinchu 30013, Taiwan
E-mail: g9562502@oz.nthu.edu.tw
E-mail: yoshijava@gmail.com
E-mail: king@cs.nthu.edu.tw
E-mail: d9562812@oz.nthu.edu.tw
*Corresponding author

**Abstract:** Many P2P VoD systems have been proposed, but only few can support VCR operations. Most of the works require very well provisioned source servers to guarantee the quality. The challenge is to do the same without requiring well provisioned source servers. In this paper, we take the heterogeneity of bandwidth into consideration, and also consider the practical case in which the capacity of the streaming buffer is limited. With these conditions, we formulate the problem as an *upload bandwidth optimisation problem*. We first prove that this problem is NP-hard and then propose an optimal dynamic programming solution.

**Keywords:** P2P; peer-to-peer; VoD; video-on-demand; VCR operation; bandwidth utilisation.

**Reference** to this paper should be made as follows: Chung, T-M., Huang, S-C., King, C-T. and Chang, C-P. (xxxx) 'Optimising upload bandwidth for quality of VCR operations in P2P VoD systems', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. x, No. x, pp.xxx–xxx.

**Biographical notes:** Tzu-Meng Chung received the BS and MS from the Department of Computer Science, National Tsing Hua University, Taiwan, ROC. His research interests include parallel and distributed processing.

Shih-Chieh Huang received the BS from the Department of Computer Science and Information Engineering, Fu Jen Catholic University, Taiwan, ROC, and the MS from the Department of Computer Science, National Tsing Hua University, Taiwan, ROC. Currently, he is pursuing his PhD in the Department of Computer Science, National Tsing Hua University, Taiwan. His current research interests include parallel and distributed processing and networked embedded systems.

Chung-Ta King received the BS in Electrical Engineering from National Taiwan University, Taiwan, ROC, in 1980, and the MS and PhD in Computer Science from Michigan State University, East Lansing, Michigan, in 1985 and 1988, respectively. From 1988 to 1990, he was an Assistant Professor of Computer and Information Science at New Jersey Institute of Technology, New Jersey. In 1990, he joined the Faculty of the Department of Computer Science, National Tsing Hua University, where he is currently a Professor and the director of the Computer and Communication Research Centre. His research interests include parallel and distributed processing and networked embedded systems. He is a member of the IEEE.

Chiu-Ping Chang received the BS and MS in Information Management from the Chaoyang University of Technology, Taiwan, ROC. Currently, she is pursuing her PhD in the Department of Computer Science, National Tsing Hua University, Taiwan. Her current research interests include distributed data processing and parallel processing.

## 1 Introduction

Peer-to-Peer (P2P) Video-on-Demand (VoD) is increasingly popular with internet users. Many P2P VoD systems have been presented (Do et al., 2004; Guo et al., 2003; Liao et al., 2006; Ying and Basu, 2005; Sheu et al., 1997; Hua et al., 1998; Zheng et al., 2006), but only few support VCR operations, such as fast-forward and jumping to arbitrary playback points (Cheng et al., 2007). The challenge lies in providing smooth playback on changes of the playback

Copyright © 200x Inderscience Enterprises Ltd.

point during VCR operations. This entails fast locating the peers that can supply the needed video blocks for the VCR operations and downloading the blocks in time.

To satisfy such a quality requirement, most previous systems rely on well-provisioned source servers. The challenge is to do the same without well-provisioned source servers. One promising approach to solving this problem is to replicate the video blocks in the overlay. BulletMedia (Vratonjic et al., 2007) uses this approach to reduce the load of the source server and the latency of resuming playback when peers perform a seek operation. However, it only considers number of replicas for a block in the overlay, but not the total available upload bandwidth for a block. Considering only number of replicas is insufficient when the upload bandwidths of the peers are heterogeneous. We address this issue in this paper.

Consider the simple example, in which there are five peers in the overlay and the video file can be divided into three blocks ($b_1$, $b_2$, $b_3$). Figure 1 shows the example with the upload bandwidth, cache content and cache capacity of each peer. For each peer, since the number of blocks is larger than the cache capacity, we must decide which block should be discarded. Figure 2 shows one solution when the peers use the 3-replica method to cache the video blocks. Although each block has three replicas in the overlay, the available upload bandwidth for each block is unbalance, e.g., the available bandwidth for $b_1$ is 7 + 3 + 5 = 15, $b_2$ is 17 and $b_3$ is 28. If the playback rate must be larger than 20 to guarantee the quality of playing the video, only block $b_3$ can provide enough upload bandwidth.

**Figure 1** An example with the upload bandwidth, cache content, and cache capacity of each peer (see online version for colours)

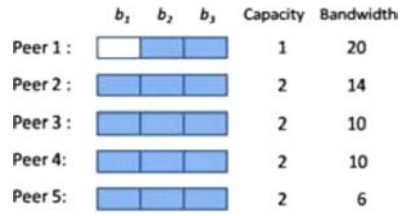
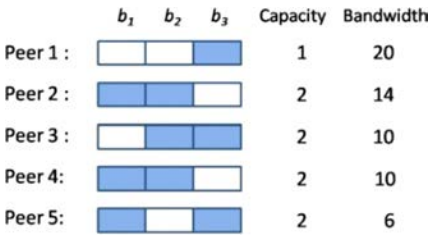
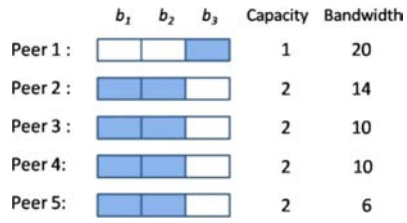



**Figure 2** A solution with the 3-replica method (see online version for colours)




Our goal of the paper is to ensure that the upload bandwidth for each block can satisfy the given bandwidth requirements. If the expected bandwidth for each block is 20 in the above-mentioned example, then one possible solution can be shown in Figure 3. In this paper, we assume that the peers have heterogeneous upload bandwidths and limited cache capacity for storing video blocks. With these constraints, the problem is to decide which block should be cached to optimise the upload bandwidth for each block. The problem is referred to as the *upload bandwidth optimisation problem* in this paper.

**Figure 3** A solution with balancing the upload bandwidth for each block (see online version for colours)




The rest of this paper is organised as follows. In Section 2, we formulate the problem and prove that the problem is NP-hard. Section 3 proposes an optimal solution using Dynamic Programming (DP). Section 4 introduces a Distributed Heuristic (DH) to solve the problem. Performance study based on simulation is presented in Section 5. Finally, we conclude this paper in Section 6.

## 2 Problem formulation

In this paper, we make the following assumptions. First, we assume heterogeneous upload bandwidth. This assumption is reasonable and consequential for heterogeneous internet. The second assumption is limited streaming cache capacity. Although the memory capacity of ordinary computers is getting larger and larger, there is still limitation on the amount that can be allocated to streaming, especially when there are other applications running at the same time. The final assumption is that there is no pre-fetching mechanism. This assumption is reasonable because not all peers have spare download bandwidth to prefetch blocks. Thus, in this paper, we only consider the replication of blocks, which have been played.

### *2.1 Model description*

We consider the states of all peers during the interval $\Delta t$ of playing the video in the P2P VoD system. Assume that the video file is decomposed into $l$ blocks and let $K$ denote the set of all blocks. Suppose further that the overlay consists of $q$ peers. Let $P$ denote the set of peers. Each peer $p$ has an upload bandwidth $u_p$ and a streaming cache capacity $c_p$.

We can define a $q \times l$ matrix $\mathbf{M}$ to describe the cache content of all peers, where $M_{p,k} = 1$, if peer $p$ has block $k$ in its cache, and $M_{p,k} = 0$ otherwise. Our problem is to decide which block should be discarded or retained to optimise the upload bandwidth. Hence, we define another $q \times l$ matrix $\mathbf{M}'$ where $M'_{p,k} = 1$, if peer $p$ has block $k$ in its cache after executing the discard strategy, and $M'_{p,k} = 0$ otherwise. The matrix $\mathbf{M}'$ is the output of our algorithm. Note that if $M_{p,k} = 0$, then $M'_{p,k}$ must be equal to 0. This is because if peer $p$ does not cache block $k$, then after executing the discard strategy, the cache content of peer $p$ will certainly

not contain block $k$. Hence, we have the following constraint:

$$M'_{p,k} \le M_{p,k} \quad \forall p \in P, \forall k \in K. \tag{1}$$

We use the following inequality to describe the constraint of limited cache capacity:

$$\sum_k M'_{p,k} \le c_p \quad \forall p \in P. \tag{2}$$

Now, let us consider the expected bandwidth $b'_k$ for block $k$. For a video file, there may be different required upload bandwidth for each block. For example, there may be some sections in a video file that are very popular. The data blocks within those sections are requested more than other blocks. Thus, these blocks require more upload bandwidth to serve the users. We can thus define the upload bandwidth $b_k$ for block $k$ as follows:

$$b_k = \sum_p \left( M'_{p,k} \times \frac{u_p}{c_p} \right). \tag{3}$$

Notations used in our formulation are summarised in Table 1. Now we demonstrate our model using the example of Figure 1. Since there are five peers and three blocks, we have $|P| = q = 5$ and $|K| = l = 3$. The upload bandwidth of the peers is $u = \{20, 14, 10, 10, 6\}$, and the cache capacity of each peer is $c = \{1, 2, 2, 2, 2\}$. Then, the cache content is expressed as follows:

$$M = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

**Table 1** Notations of our formulation

| *Notation* | *Definition* |
|---|---|
| $P$ | The set of peers |
| $K$ | The set of blocks of video |
| $q$ | The number of peers |
| $l$ | The number of blocks |
| $u = \{u_p, p \in P\}$ | Upload bandwidth of peers |
| $c = \{c_p, p \in P\}$ | Cache capacity of peers |
| $b = \{b_k, k \in K\}$ | Bandwidth that block $k$ is allocated |
| $b' = \{b'_k, k \in K\}$ | Bandwidth that block $k$ requires |
| $M = (M_{p,k})_{q \times l}$ | Initial cache content matrix |
| $M' = (M'_{p,k})_{q \times l}$ | Final cache content matrix |

We assume the expected upload bandwidth for each block is uniform and assign them the average bandwidth $\left(= \Sigma_p u_p / l\right)$. Hence, the expected bandwidth for each block is $b' = \{20, 20, 20\}$. Figure 2 shows a solution using the 3-replica method, which can be expressed as follows:

$$M' = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

Moreover, the bandwidth that each block can get is obtained by the following calculations:

$$b = \left(\frac{u}{c}\right) \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$$= \left( \frac{20}{1} \quad \frac{14}{2} \quad \frac{10}{2} \quad \frac{10}{2} \quad \frac{6}{2} \right) \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 15 & 17 & 28 \end{pmatrix}.$$

## *2.2 Formulation*

Given the current cache contents of the peers in the P2P VOD system and the blocks just played, the upload bandwidth optimisation problem is to decide which blocks should be discarded/retained in the cache of the peers so that the upload bandwidth of each block can satisfy the given bandwidth requirements. Using the constraints given in the previous subsection, we can now formulate our problem as an optimisation problem with the following objective:

$$P: \text{minimise} \quad \sum_{k \in K} (b_k - b'_k) / 2. \tag{4}$$

The optimal solution is the one that allocates the bandwidth to the blocks as close to the required bandwidth as possible. We use the square of the difference between $b$ and $b'$ to represent our optimal goal. In the example of Figure 2, we can calculate the objective function as follows:

$$\sum_{k \in K} (b_k - b'_k)^2 = (15-20)^2 + (17-20)^2 + (28-20)^2 = 98.$$

The objective function of the example in Figure 3 is given here:

$$\sum_{k \in K} (b_k - b'_k)^2 = (20-20)^2 + (20-20)^2 + (20-20)^2 = 0.$$

Hence, we can say that the performance of Figure 3 is better than Figure 2.

## *2.3 NP-hard*

In this subsection, we show that the upload bandwidth optimisation problem is NP-hard. Our proof is via the *subset sum problem*, which is known to be NP-complete. The input

to the subset sum problem is a set of $n$ integers and an integer $s$. The output is 'yes' if and only if the sum of some non-empty subset is equal exactly to $s$.

**Theorem 1:** *The upload bandwidth optimisation problem is NP-hard.*

*Proof*: We consider the case of $l = 2$ of the optimising bandwidth utilisation problem. Let the cache capacity and the cache content of each peer be as follows:

$$cp = 1 \quad \text{for } \forall\, p \in P \tag{5}$$

$$M = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ \vdots & \vdots \\ 1 & 1 \end{pmatrix}. \tag{6}$$

Equation (6) indicates that all peers have cached $b_1$ and $b_2$. Equation (5) means that each peer only can cache one block. Thus, each peer must decide which block should be discarded to achieve our goal in equation (4).

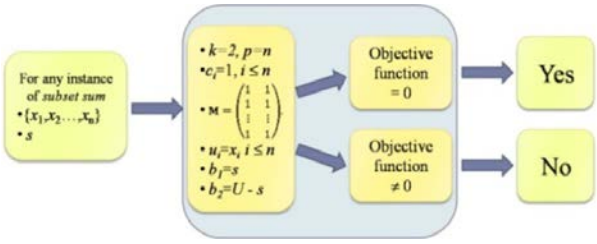
Let $\{x_1, x_2, \ldots, x_n\}$ and $s$ define the instance of the subset sum problem. We let the number of the peers $q = n$, the bandwidth $b_i = x_i$ for each peer $i$ and the expected bandwidth $b'_1 = s$ and $b_2 = U - s$, where $U = \sum u_i$. Hence, the optimal objective equation (4) can be expressed as follows:

$$\sum_{k \in K} (b_k - b'_k)^2 = (b_1 - s)^2 + (b_2 - (U - s))^2. \tag{7}$$

If the objective function (7) equal to zero, $b_1$ must be equal to $s$. That is to say, we can find a set of peers that the sum of their bandwidths is equal to $s$. Hence, the answer to the given instance of the subset sum problem is 'yes' if the corresponding objective function of the upload bandwidth optimisation problem equals to zero.

An illustration of the transformation is shown in Figure 4. Since the transformation takes polynomial time, a polynomial time algorithm for the upload bandwidth optimisation problem would imply a polynomial time algorithm for the subset sum problem. Hence, the upload bandwidth optimisation problem is NP-hard.

**Figure 4** The transformation of the subset sum problem to the upload bandwidth optimisation problem (see online version for colours)




## 3 Optimal DP solution

In this section, we propose an optimal DP solution. For the subset sum problem, there is a pseudo-polynomial time DP solution. We have proven that the subset sum problem is a special case of our problem in the previous section. Thus, we can find an optimal DP solution to the upload bandwidth optimisation problem using the same idea.

The simple idea for solving the subset sum problem is to use a one-dimensional Boolean array to record the possible sum. For our problem, we can also use an $l$-dimensional Boolean array to record the possible upload bandwidth for each block. Hence, we can define an $l$-dimensional function $D_n(X)$, where $X = (x_1, x_2, \ldots, x_l)$. Note that $D_n(X) = 1$ if $(x_1, x_2, \ldots, x_l)$ is the possible upload bandwidth for the blocks $(b_1, b_2, \ldots, b_l)$, when $n$ peers have already executed the discard strategy. Otherwise, $D_n(X) = 0$.

The recursive formula is expressed as follows:

$$D_0 = b''_1, b''_2, \ldots, b''_l = 1, \quad \text{for } b''_k = \sum_p \left( M_{p,k} \times \frac{u_p}{c_p} \right). \tag{8}$$

In equation (8), $b''_k$ represents the upload bandwidth for block $k$ before all peers execute the discard strategy. Hence, $D_0 = b''_1, b''_2, \ldots, b''_l = 1$. In the example of Figure 1,

$b''$ = (14/2 + 10/2 + 10/2 + 6/2,
14/2 + 10/2 + 10/2 + 6/2+ 20/1,
14/2 + 10/2 + 10/2 + 6/2 + 20/1) = (20, 40, 40).

Hence, $D_0$ (20, 40, 40) = 1 initially.

In equation (9), we assume that we have known the upload bandwidth for each block after peers 1, 2, …, $i - 1$ execute the discard strategy. Next, we calculate the possible upload bandwidth when peer $i$ executes the discard strategy. If peer $i$ has block $b_1$ ($M_{i,1} = 1$) and decide to discard $b_1$, then the upload bandwidth $(x_1, x_2, \ldots, x_l)$ can be calculated from the term $(D_{i-1}(x_1 + u_1/c_1, x_2, \ldots, x_l)$ AND $M_{i,1}$, and so on. Hence, $D_n(X)$ records all possible answers finally. Thus, we can find the optimal solution from

$$\min \left\{ \sum_{k \in K} (x_k - b'_k)^2 \mid D_n(x_1, x_2, \ldots, x_l) = 1 \right\}.$$

Although in theory we can solve the problem using DP, it could not be possible in practice. There are two main reasons. One is that the P2P VoD system is a distributed environment but the DP solution is a centralised algorithm. The other is that the time and space complexity of DP solution is $O(V^k)$, where $V$ is the possible value of upload bandwidth. The time and space complexity is too high for practical use. Thus, we propose a Distributed Heuristic (DH) in the next section.

## 4 Distributed Heuristic

A P2P VoD system is a distributed system. One fundamental problem about distributed environment is how to get the information from other peers. There are some techniques to achieve this, such as gossiping (Ganesh et al., 2003; Zhang et al., 2005), DHT and clustering. However, gossiping may not be efficient for P2P VoD systems

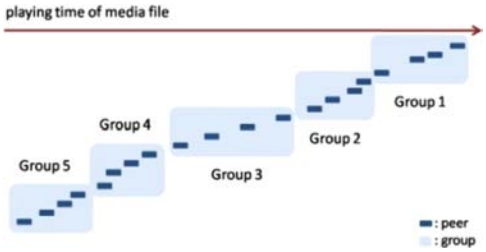
because peers usually have different playback points across a wide range. For DHT, there is heavy overhead in lookup and update since we must decide the discarded block after playing each block. Thus, we decide to follow the clustering strategy and to divide the peers into groups to get the information.

Our heuristic consists of three stages: group management, managed range decision and greedy discard algorithm.

### *4.1 Group management*

As shown in Figure 5, we divide the peers into groups by their playback points. The reason for doing this is that peers close in their playback points may have similar data blocks. This makes our discard strategy simpler. We can assign the group head as the peer, which has the latest playback point. When a peer joins a group, it contacts the group head and gets group information. When the peer leaves, it also notifies the group head. Thus, the group head has the newest group information and each peer can refresh the information from the head. The group information includes the cache content, cache capacity, upload bandwidth and playback point of the peers.

**Figure 5** The peers are divided into groups by their playback points (see online version for colours)

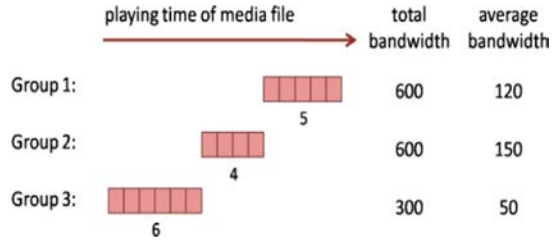



Now, let us consider how to refresh the group information with a low overhead. Note that peers not only play the video sequentially but also perform VCR operations. Fortunately, sequential playback occupies most of the time. Thus, the basic idea of our heuristic is to let the peer simulate the behaviour of other peers locally assuming that they are sequentially playing. This significantly reduces the frequency for peers to refresh the group information. The peers only refresh the group information from the head after a constant time *t*.

### *4.2 Managed range decision*

In the second stage, we decide the range of blocks to be managed by each group. Since the playback points of the peers within the same group span a certain range, the peers need not consider those blocks far away from the assigned range. The simplest idea to decide the range to be managed by each group is the playback range of all peers within the same group. However, this may make the upload bandwidth of each group unbalanced. For example, as shown in Figure 6, the managed range and the total bandwidth of each group are different. We assume the expected bandwidth for each block is the average bandwidth ((600 + 600 + 300)/15 = 100). Now even if each group can ensure its own managed blocks to have the balance bandwidth, e.g., group 1 has bandwidth 120 and group 2 has bandwidth 150, the bandwidths across the groups are uneven, while the ideal allocated bandwidth for each block is 100. Thus, we must decide a better range for each group to manage.

**Figure 6** An example of unbalance bandwidth of each group (see online version for colours)




One simple idea to solve the above problem is to let the group that has insufficient bandwidth borrow the bandwidth from the group, which has spare bandwidth. Figure 7 shows the idea for solving the problem of Figure 6. Group 3 only has the total bandwidth 300 and its managed range has six blocks. Thus, it borrows the upload bandwidth 300 from group 2. In other words, group 2 needs to manage additional four blocks by using the quota of bandwidth 300. Similarly, group 2 borrows a bandwidth of 100 from group 1. Hence, each block can achieve the expected bandwidth.

**Figure 7** The basic idea for solving the unbalance bandwidth problem (see online version for colours)

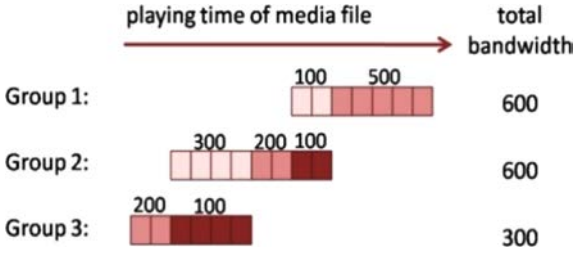



Figure 8 shows the pseudo-code of calculating the expected bandwidth and managed range.

### *4.3 Greedy discard algorithm*

In the first two stages, we already have obtained the group information and decided the managed range of each group. Thus, the peers only need to execute a localised algorithm using this information to decide which block should be discarded. We use a greedy algorithm to decide the block to be discarded. First, we calculate the total upload bandwidth of each block when all peers have not discarded any block. Second, we calculate all the objective functions when a block is discarded in the peer's cache. The block that has the minimal objective function will be discarded. Figure 9 shows the pseudo-code of our greedy algorithm.

**Figure 8** The pseudo-code for deciding the managed range for each group

**Input:**

*num_block* : number of blocks of the video;
*left_band*: total bandwidth of other groups managing blocks with earlier playback time;
*right_band*: total bandwidth of other groups managing blocks with later playback time;
*group_band*: total bandwidth of the group;
*group_start*: the start playback offset of the group;
*group_end*: the end playback offset of the group;
*ratio*: the ratio of bandwidth between two groups.

**Manage Range Decision:**

```
expect_band ← (left_band + right_band + group_band) / num_block;
left_exc_band ← expect_band *(group_start – 1);
right_exc_band ← expect_band*(num_block – group_end);

for i = group_start to group_end then
  exc_band[i] ← expect_band;
end for i;

left_rem_band ← left_exc_band - left_band;
right_rem_band ← right_exc_band - right_band;

if left_rem_band < 0 then
  L ← (-left_rem_band )/ (ratio * expect_band);
  for i = group_start –L to group_start-1 then
    exc_band[i] ←ratio * expect_band;
  end for i;
end if;

if right_rem_band > 0 then
  R ← right_rem_band / (ratio * expect_band);
  for i = group_end – R to group_end then
    exc_band[i] ← (1 - ratio) * expect_band;
  end for i;
end if;
```

**Output:**

*exc_band*[*i*] : expected bandwidth of block *i*.

**Figure 9** The pseudo-code of the local greedy discard algorithm

**Input:**

*num_member* : number of peers of the group;
*band*[*i*] : upload bandwidth of peer *i*;
*cache_cp*[*i*] : cache capacity of peer *i*;
*cache_content*[*i, j*] : cache content of peer *i*;
*exc_band* [*i*] : expected bandwidth of block *i*;
*set_member* : set of members of the group;
*set_block* : set of blocks managed by group.

**Greedy-Algorithm:**

```
for block j ∈ set_block do
    B[j] ← 0;
    for peer i ∈ set_member do
      if cache_content[i,j] = true then
        B[j] ← B[j] + band[i] /cache_cp[i];
      end if;
    end for i;
    V[j] ← (B[j] –exc_band [j])* (B[j] –exc_band
    [j]);
end for j;

for peer i ∈ set_member do
    now_value ← -1;
    for block j ∈ set_block do
      if cache_content[i,j] = true then
        temp_value←(B[j]-band[i]/cache_cp[i]-
        exc_band [j]);
        temp_value ← temp_value * temp_value;
        if (V[j] - temp_value) > now_value or
        now_value = -1 then
          now_value ← V[j] - temp_value;
          discard_block ← j;
        end if;
      end if;
    end for j;

cache_content[i, discard_block] ← false;
B[j] ← (B[j] – band[i] /cache_cp[i]);
V[j] ← (B[j] – exc_blo[j])* (B[j] –exc_band [j]);
end for i;
```

**Output:**

*cache_content*[*i, j*].

## 5 Evaluation

In this section, we compare our heuristic and the replica method in Vratonjic et al. (2007) and verify two points using simulation. One point is that the replica method is unsuitable when the upload bandwidth is heterogeneous. The other is that our objective function actually conforms to our optimisation goal.

### *5.1 Experimental set-up*

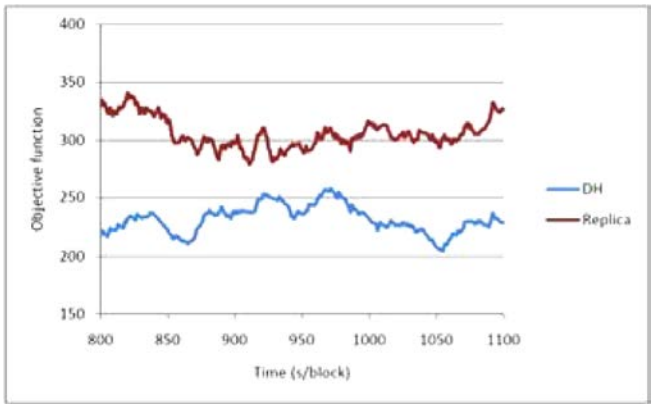
For all our experiments, we set the upload bandwidth of all peers to have four types: 100 kB/s, 200 kB/s, 1000 kB/s and 2000 kB/s. The join model of the peers is according to a Poisson process. All peers watch video from the beginning. We allow the peers to perform VCR operations to jump to a random part of the video file. There are roughly 500 peers to watch video at the same time.

The size of the video file is 600 blocks. The expected bandwidth for each block is according to a uniform distribution with a mean value equal to average upload bandwidth. In other words, the upload bandwidth for each block is balanced. In our experiments, we assume one unit time is the time to play one block. We observe our objective function value and the success rate of fetching blocks for a period of time.

## *5.2 Performance evaluation*

We first compare our DH with the replica method (Vratonjic et al., 2007) by our objective function. We normalise our objective function as $\left(\sum_{k\in I}(b_k - b'_k)^2 / K\right)^{1/2}$. The normalisation is to express the difference between the allocated bandwidth and the expected bandwidth per block. Figure 10 shows the result. Our heuristic is always better than the replica method, since the replica method does not consider the heterogeneous bandwidth.

**Figure10** Comparison of the Distributed Heuristic (DH) with the replica method (Vratonjic et al., 2007) (Replica) (see online version for colours)




Next, we compare our DH with the replica method by the success rate of fetching blocks. We assume that the playback rate must be larger than 100 kB/s to ensure the quality of watching the video. Thus, if the peer can fetch the playback block from other peers, we say this is a successful fetching behaviour. Otherwise, the peer must fetch the block from the source server. Figure 11 shows the result. Our DH always has better performance than the replica method. The replica method increases the load of the source server. In both Figures 10 and 11, the results also verify that our objective function actually conforms to our optimisation goal. The value of the objective function using our DH is smaller and the load of the source server is also less.

## *5.3 Effects of group size and refresh time*

Our heuristic has two important parameters: group size and refresh time. In this section, we discuss the effects of these parameters.

Figure 12 shows the effects of different group sizes. The performance is almost the same with different group sizes. This is because the peers of the VoD system usually have different playback points across a wide range. If the group size is large, then the playback range is also large. In other words, the performance is affected only by the peers that have close playback points.

**Figure 11** The success rate of fetching blocks (see online version for colours)

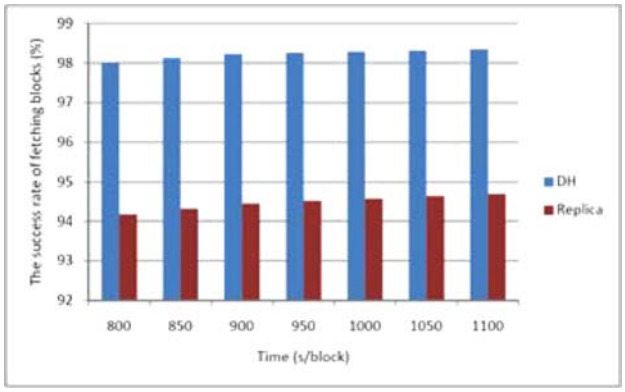



**Figure 12** Comparison of different group size (see online version for colours)

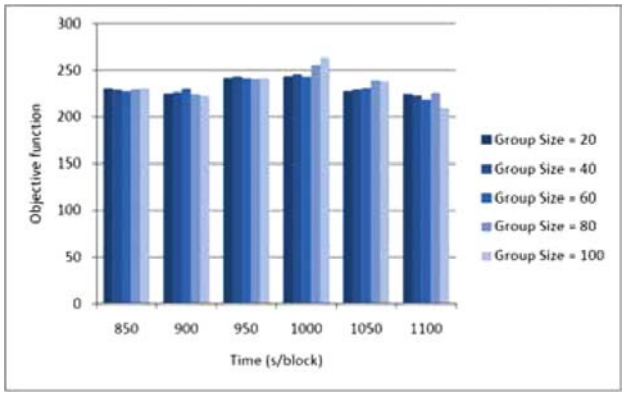
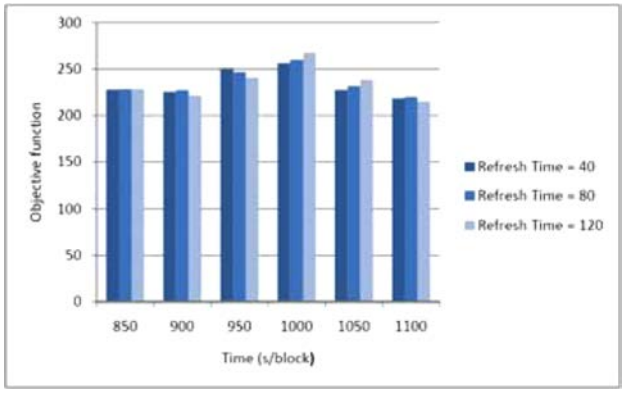



Figure 13 shows the result of different refresh time. As expected, the different refresh times have almost the same performance. The reason is that we have the peer simulation mechanism in our heuristic. Since the group size and refresh time does not have obvious effects on the performances, our DH can achieve low overhead by setting a small group size and long refresh time.

**Figure 13** Comparison of different refresh time (see online version for colours)



## 6 Conclusion

In this paper, we formulate the bandwidth utilisation problem on the P2P VoD systems that provide VCR operations, and show this problem is NP-hard. We propose an optimal DP solution to solve this problem and a DH to work in practice. Our simulation results show that our formulation actually conforms to our optimisation goal, which is to guarantee the quality of VCR operations without requiring a well-provisioned server. Our distributed heuristic can achieve low overhead by setting a small group size and long refresh time and has better performance than previous work using the replica method.

## Acknowledgement

This work was supported in part by the National Science Council, Taiwan (ROC), under grant NSC 96-2218-E-007-009, the Ministry of Economic Affairs, Taiwan (R.O.C.), under grant 96-EC-17-A-04-S1-044, and the Industrial Technology Research Institute, Taiwan (R.O.C.), under grant G1-97005.

## References

Cheng, B., Jin, H. and Liao, X. (2007) 'Supporting VCR functions in P2P VoD services using ring-assisted overlays', *Proc. IEEE International Conference on Communications (ICC)*, 24–28 June, Glasgow, Scotland, pp.1698–1703.

Do, T.T., Hua, K.A. and Tantaoui, M.A. (2004) 'P2VoD: providing fault tolerant video-on-demand streaming in peer-to-peer environment', *Proc. IEEE International Conference on Communications (ICC)*, 20–24 June, New York, USA, pp.1467–1472.

Ganesh, J.A., Kermarrec, A.M. and Massoulie, L. (2003) 'Peer-to-peer membership management for gossip-based protocols', *IEEE Trans. on Computers*, Vol. 52, No. 2, February, pp.139–149.

Guo, Y., Suh, K., Kurose, J. and Towsley, D. (2003) 'P2Cast: peer-to-peer patching scheme for VoD service', *Proc. 12th International Conference on World Wide Web*, 20–24 May, Budapest, Hungary, pp.301–309.

Hua, K.A., Cai, Y. and Sheu, S. (1998) 'Patching: a multicast technique for true video-on-demand services', *Proc. ACM International Conference on Multimedia*, 12–16 September, Bristol, UK, pp.191–200.

Liao, C.S., Sun, W.H., King, C.T. and Hsiao, H.C. (2006) 'OBN: peering for finding suppliers in P2P on-demand streaming systems', *Proc. IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, 12–15 July, Minneapolis, USA, pp.235–242.

Sheu, S., Hua, K. and Tavanapong, W. (1997) 'Chaining: a generalized batching technique for video-on-demand systems', *Proc. IEEE International Conference on Multimedia Computing and Systems (ICMCS)*, 3–6 June, Ottawa, Canada, pp.110–117.

Vratonjic, N., Gupta, P., Knezevic, N., Kostic, D. and Rowstron, A. (2007) 'Enabling DVD-like features in P2P video-on-demand systems', *Proc. ACM SIGCOMM on Peer-to-Peer Streaming and IP-TV Workshop (P2P-TV)*, 27–31 August, Kyoto, Japan, pp.329–334.

Ying, L.H. and Basu, A.(2005) 'pcVOD: internet peer-to-peer video-on-demand with storage caching on peers', *Proc. 11th International Conference on Distributed Multimedia Systems (DMS)*, 5–7 September, Banff, Canada, pp.218–223.

Zhang, X., Liu, J., Li, B. and Yum, T-S.P. (2005) 'CoolStreaming/DONet: a data-driven overlay network for live media streaming', *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, 13–17 March, Miami, USA, pp.2102–2111.

Zheng, C., Shen, G., Li, S. and Shenker, S. (2006) 'Distributed segment tree: support of range query and cover query over DHT', *Proc. International Workshop on Peer-to-Peer Systems (IPTPS)*, 27–28 February, Santa Barbara, USA, pp.5–15.