A Fast SVC-Based Channel-Recommendation System for an IPTV on a Cloud and P2P Hybrid Platform

HONG-YI CHANG1,∗, CHIH-CHUN LAI1 and YUAN-WEI LIN2

1Department of Management Information Systems, National Chiayi University, Chiayi, Taiwan
2Department of Mobile Internet, Industrial Technology Research Institute, Hsinchu, Taiwan
∗Corresponding author: hychang@mail.nctu.edu.tw

In this paper, we propose a fast scalable video coding (SVC)-based channel-recommendation system for IPTV on a Cloud and peer-to-peer (P2P) hybrid platform. When a user switches channels, the system redirects the client to the cloud network and delivers the base layer of SVC streams to the client. The system provides a multichannel preview window with a small resolution and a fast channel-switching mechanism without additional traffic. After a user has selected a channel, the system redirects the client to the P2P network and sends the necessary enhancement layer streams, so that the client can receive high-quality video. Because of the fact that the recommendation system is known as an effective approach for enhancing the efficiency of channel previews, we propose a novel recommendation system based on the feedback loser tree (FLT) algorithm. The FLT algorithm can be trained by the user’s historical log, and attempt to find the user’s preferred channels quickly. Our experimental results indicate that the proposed platform can obtain a higher peak signal to noise ratio quality than the original P2P networks, and the proposed system can help users find their favorite channels in only 2 to 5 switching pages. The performance of the proposed system is ∼75% better than that of the high-performance multichannel preview system.

Keywords: peer-to-peer; cloud computing; scalable video coding; Internet TV; loser tree; channel recommendation system

Received 25 January 2013; revised 30 June 2013
Handling editor: Yannis Manolopoulos

1. INTRODUCTION

Peer-to-peer (P2P) technology has made the concept of resource-sharing a reality. In earlier years, only small files such as plain text or audio files could be transmitted using P2P. However, with the growth of this technology, large multimedia data can now be transmitted via P2P. In addition, the increasing demand for Internet services has triggered a rise in Internet TV (IPTV) services. Users can now watch TV programs through the Internet with a mechanism that allows multimedia streams to flow through a P2P network. Many popular products are available, such as [1–4], to make this type of entertainment available.

There are many factors that have led to traditional TV still not being replaced by IPTV, wherein there are two factors that are most important. The first, any peer of a P2P network may join or leave the network unpredictably, causing P2P instability. If the peer who is providing streams to other clients leaves or crashes, the provided program breaks off because it no longer has stream resources. At this time, the served clients must search for another peer to provide streaming, and this means that they must watch the program intermittently. Setting a huge stream buffer for each client to avoid program breaks when other peers leave is currently a popular solution, but this causes that clients are required to wait for some time to watch a program after they change channels. The main reason is that the media player continues consuming the previously buffered stream after clients change channels, and the new program is played until the buffered stream of this program suffices to combat the instability of the P2P network. The long delay time when
changing channels caused by a ‘huge buffer’ mechanism is an extreme nuisance to IPTV users. This is an urgent problem to be addressed for IPTV. In this paper, we call this the ‘Hanging Time between Channels’ (HTC) problem of IPTV. The second, even the system has a mechanism for multichannel preview, and users still need to spend a lot of time to switch channels. Therefore, an efficient channel recommendation system needs to be proposed for users to find their favorite channels quickly.

In order to find an effective solution for HTC, we proposed a ‘Cloud to P2P’ (C2P) IPTV platform in our previous work [5] that integrates cloud technology [6]. The high scalability and high stability of cloud servers to support fast channel switching of huge number of concurrent users will reduce the delay time, provide channel previews by scalable video coding (SVC) technology, and realize the seamless channel-switching mechanism of IPTV. However, it is possible to reduce the channel-switching time. This paper proposes an efficient channel recommendation system and makes the following contributions:

(i) We propose a novel recommendation system based on the feedback loser tree (FLT) algorithm. The FLT algorithm can be trained by the users’ historical log, and attempt to find the users’ preferred channels quickly. The recommended results of the FLT are integrated in the preview pages of the C2P platform.

(ii) The proposed system can help users find their favorite channels in only 2 to 5 switching pages. The performance of the proposed system is \( \sim 75\% \) better than that of the high-performance multichannel preview (HPMP) system.

(iii) The FLT algorithm is compared with the user-based collaborative Filtering algorithm using the Pearson correlation coefficient (UPCC) [7] and the PageRank (PR) [8] based on the MovieLens data sets. The results show that the predictive accuracy of the FLT is better than that of the UPCC and PR algorithms.

The rest of this paper is organized as follows: Section 2 presents a review of the related works; Section 3 introduces the proposed designs for C2P architecture and the FLT method. Section 4 shows the simulation and analysis. Finally, Section 5 offers a conclusion.

2. RELATED WORKS

2.1. A survey of channel-switching acceleration methods

Past researchers have proposed channel-switching acceleration methods such as multichannel delivery methods [9]. This method simultaneously delivers multiple channels and predicts the next channel that the user wants by using user behavior.

As shown in Fig. 1, to reduce HTC, all of the predicted channel streams are delivered with the stream the user is watching. However, this causes more bandwidth consumption because multiple channel streams are transmitted simultaneously.

The other solution is to reduce the transmission delay between peers. Figure 2 shows the concept of Smart-Fit [10], which can predict user behavior by using a decision tree, and to dynamically set the next channel on the location of the decision

![Figure 1](image1.png)

**FIGURE 1.** Deliver multiple channel streams to reduce HTC.

![Figure 2](image2.png)

**FIGURE 2.** The example of building the template.
A Fast SVC-Based Channel-Recommendation System

When a user is watching a program, the Smart-Fit generates the next recommended channel for this user, and provides the streams when the user changes to this channel. However, a delay of more than 17% exists between this method and the optimal solution.

A CAPS [11] algorithm can distribute bandwidth and provide an efficient peer selection method. Figure 3 shows that some peers would be covered by multiple views of channels. How to choose a peer with sufficient bandwidth to establish the overlay is a crucial problem. The CAPS algorithm has been proved to be more effective than a traditional random peer selection (RPS) algorithm in solving peer selection problems. However, its performance could still be improved in a multi-view environment.

A HPMP [12] system employs the multichannel preview function that enables users to watch multiple channels simultaneously and improves the performance of the channel preview. The authors designed a model to calculate the quality \( Q \) of each watch peer and the stability \( R \) of each preview peer, to attempt to identify stable source peers and select certain leader peers among them. Thereafter, the leader peers would choose multiple channels from the HPMP system and deliver streams to tree leaves. Figure 4 shows the main concept of their timed forest scheme (TFS) algorithm.

Although the HPMP system can efficiently reduce the channel-hopping time and lessen the load of tracker servers, the method still has certain drawbacks (e.g. if a user wants to watch a specific channel but the quality of the source peers is poor, such as an insufficient number of source peers or frequently joining or leaving the system). A channel including poor-quality source peers obtains a low \( Q \) value after calculation, and that results in users being unable to preview the channel quickly.

2.2. A survey of recommendation systems

The various approaches to Recommendation Systems can be broadly categorized as follows:

1. Collaborative filtering (CF): In CF systems, a user is recommended items based on the past collective ratings of all users. CF methods can be subdivided into memory-based and model-based approaches.

   a. Memory-based CF: Memory-based methods are also commonly referred to as neighborhood-based approaches. A subset of users is chosen based on their similarity to the active user, and a weighted combination of their ratings is used to produce predictions for this user. The most commonly used measure of similarity is the Pearson correlation coefficient (PCC) [7] between the ratings of two users. Memory-based CF can be subdivided into user-based [7] and item-based [13] approaches. A user-rating matrix is shown in Fig. 5 to illustrate the differences between the two methods. The user-based approach calculates the similarity between the horizontal line data values of the two users. For example, the \( PCC_{sim}(u_a, u_n) \) is a measure of the similarity between user \( n \) and active user \( a \). The PCC

![FIGURE 3. Overlay of multiple channels in P2P networks.](image)

![FIGURE 4. TFS algorithm.](image)
between the ratings of the two users is defined below:

\[
PCC_{\text{Sim}}(u_a, u_n) = \frac{\sum_{i \in I} (r_{a,i} - \bar{r}_a) - (r_{n,i} - \bar{r}_n)}{\sqrt{\sum_{i \in I} (r_{a,i} - \bar{r}_a)^2 \sum_{i \in I} (r_{n,i} - \bar{r}_n)^2}},
\]

(1)

where \( I \) is the set of items rated by both users, \( r_{n,i} \) is the rating given to item \( i \) by user \( n \) and \( \bar{r}_n \) is the mean rating given by user \( n \).

The item-based approach calculates the similarity between the vertical line data values of the two items. The \( PCC_{\text{Sim}}(i_j, i_m) \) is a measure of the similarity between the pairs of items \( j \) and \( m \) are computed offline using PCC, and are given by

\[
PCC_{\text{Sim}}(i_j, i_m) = \frac{\sum_{u \in U} (r_{u,j} - \bar{r}_j) - (r_{u,m} - \bar{r}_m)}{\sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2 \sum_{u \in U} (r_{u,m} - \bar{r}_m)^2}},
\]

(2)

where \( U \) is the set of all users who have rated both items \( j \) and \( m \), \( r_{u,j} \) is the rating of user \( u \) on item \( j \) and \( \bar{r}_j \) is the average rating of the \( j \)th item across users.

Deshpande et al. [14] showed that the computational complexities of the item-based approach with PCC (IPCC) and the user-based approach with PCC (UPCC) are both \( O(m^2 n) \) as we need to compute \( m(m - 1) \) similarities, each potentially requiring \( n \) operations.

(b) Model-based CF [15]: The user-based CF and item-based CF in memory-based CF have a common disadvantage: data sparsity. The enormous amount of data is not easy to handle, and consequently, the analysis results are affected. Therefore, a model-based CF was developed to solve this problem. First, the model-based CF collects the historical data of users and constructs a model, and the predictions from this model are then recommended to the target customers. Model-based CF often used techniques, such as the Bayesian model, cluster models and an artificial neural network (ANN).

(2) Content-based (CB) recommendations [16]: These approaches recommend items that are similar in content to items that the user has liked in the past or are matched to the attributes of the user. First, the user profile is compared with the characteristics of the content property to determine the relationship between them. The results are then compared with the interests and browsing behavior of the customer again. Finally, the potential items that customers may be interested in are determined, and items are recommended to the customer.

(3) Hybrid approaches: These methods combine the collaborative and content-based approaches. In general, hybrid approaches could improve certain problems in the collaborative and content-based approaches, such as the content-boosted CF algorithm [17], and are also helpful in addressing the sparsity problem in which external content information can be used to produce predictions for new users or new items.

(4) PageRank (PR) [8]: This method is a numerical value that represents how important a web page is on the web. For example, Google assumes that when one web page links
to another page, it is effectively casting a vote for the other page. The more votes that are cast for a web page, the more important the page must be. In addition, the importance of the page that is casting the vote determines the importance of the vote itself. Google calculates a web page’s importance from the votes that are cast for it.

In [18], the authors assumed that the social network has $n$ nodes, and $m$ adversarially chosen edges arrive in a random order. The authors showed that with a reset probability of $\varepsilon$, the expected total work needed to maintain an accurate estimate (using the Monte Carlo method) of the PageRank of every node at all times is in $O(n \ln m/\varepsilon^2)$.

(5) Multimedia recommender: Albanese et al. [19] presented a novel multimedia recommender based on object features and user behaviors. The method uses the user’s low-level features and the semantic descriptors to predict the user behavior and produce effective recommendations. The authors proposed the concept of multichannel browser that the user can navigate multiple media channels at the same time by only a browser.

Figure 6 shows the system architecture of the multimedia recommender. In the front end of the system, users can access various media channels through the multichannel browser. When a user is browsing the channels, the system will record the requested items to the usage log. At the same time, the pattern discovery subsystem will try to classify the user and generate the user preference.

Albanese et al. [20] proposed a novel approach to be recommended for multimedia objects, based on an ‘importance ranking’ algorithm that highly resembles the well-known PageRank ranking strategy.

3. THE C2P PLATFORM AND FLT RECOMMENDATION MECHANISM

3.1. The C2P platform with SVC

In this paper, cloud computing was integrated with P2P networking to propose a cloud IPTV platform, which can reduce HTC and realize seamless channel switching of IPTV by introducing SVC [21–23] technology. Figure 7 illustrates the architecture of the proposed cloud IPTV platform, which is composed of five major parts.

3.1.1. Preview mode

Base layer frames of multiple channels are delivered to the user in preview mode through cloud networks, offering them numerous choices. In this mode, the advantages of high computing power and high scalability of cloud computing are used for the fast transmission of multiple base layer streams to the user.

3.1.2. Cloud networks

This part contains three types of components: a cloud management server (CMS), a cloud backup pool (CBP) and cloud servers (CSs). A CMS is responsible for booting and shutting down CSs, recording information and loading CSs and optimizing the amount of running CSs by dynamic dispatching and a synchronization mechanism. A CBP can store and manage temporarily unused CSs. When a CS running in the cloud IPTV platform crashes, it is immediately replaced by a CS in a CBP. The task of the CS is to store various types of multimedia data and to deliver streams to clients.

3.1.3. Seamless channel-switching mechanism

The original video was encoded with an SVC encoder, and generates various streams belonging to different layers.
The SVC extractor sends the base layer streams to cloud networks to make multiple-channel previews in preview mode, and simultaneously sends base layer and enhancement layer streams to P2P networks, so that when the user chooses a channel, the base layer and enhancement layer streams are delivered to the user via P2P for saving cloud cost expenditure. Taking Fig. 8 as an example, the user enters the preview mode after logging into the platform, the system requests the CMS to deliver \( n \) base layer streams (\( n \) equals 4 in this example) to this user through cloud networks and the streams are stored in the client buffer. At this time, the user can concurrently watch \( n \) channel previews.

In the preview mode, the user can either choose a channel to watch or navigate to the next page of channel previews. If the user chooses the next page, the system sends a request to the CMS again to deliver the next \( n \) base layer streams to this client. If the user chooses a channel, the system notifies the P2PMS regarding the chosen channel, and the P2PMS then conducts a search and guides the user to obtain enhancement layer streams from P2P networks.

In the watch mode, assuming that the time the client switches channels to watch mode is \( t \), the client replicates \( n \) base layer streams of the preview mode in the client buffer to the base layer buffer in time \( t \). When the user wants to switch to the preview mode from the watch mode again, the client sends a signal to the CMS and requests to receive the preview-mode streams before the client switches to the review mode. The client plays preview-mode streams from the base layer buffer, saving \( n \) base layer streams in the base layer buffer at time \( t \). The CMS transmits \( n \) base layer streams at time \( t + 1 \) to the client.
With this mechanism, the user can quickly and ‘seamlessly’ switch between the preview mode and the watch mode.

3.1.4. P2P networks
After the user chooses the desired channel, the user is switched to the P2P delivery mode, and the enhancement layers of this program are delivered through P2P networking. A P2P management server (P2PMS) can find peers who are able to provide these streams quickly, and the user could obtain the enhancement layer frames from these peers via P2P.

3.1.5. Watch mode
The user can select a channel from the preview mode, and watch this channel in high quality. If the user switches back to the preview mode, only the base layer frames are played, meaning that it is a low-resolution preview video.

3.2. The FLT recommendation mechanism
To enable users to search for their favorite channels quickly, improving the architecture of IPTV does not suffice. We expect our platform to achieve the goals of user customization such as multimedia recommenders \[19, 20\]. Therefore, we propose an efficient mechanism of channel recommendations by using user logs to train and establish the preview tree (P-tree) for each user. Thereafter, we use this P-tree to identify the recommendation value \(R_v\), which is the channel preference of each user. A higher \(R_v\) means that the channel has more of a chance to be recommended in the front pages. Figure 9 shows the FLT mechanism. The recommended results of the FLT are integrated in the preview page of the C2P platform.

When a user logs into the C2P platform, the system sends a request to fetch the P-tree from the preview database, which stores a large number of P-trees of each user. The corresponding P-tree is delivered to the recommendation module (RM) for recommending the first four channels by the FLT algorithm. Thereafter, the user enters the preview mode (P-mode) and previews multiple channels. At this time, the user has three options to operate:

1. **Switch to next/backward page.** If the user switches pages, the system requests the RM to recommend the next or previous page.
2. **Mark channels.** The user might be interested in certain channels when previewing, but the user chooses not to watch them at this time. Therefore, we provide the mark module (MM) to record user-marked channels to a mark list and to dynamically adjust the \(R_v\) of each channel, allowing users to find channels that interest them quickly.
3. **Select channel to watch.** If the user finds a favorite channel, the user can click the channel and switch to the watch mode (W-mode) instantly.

If the user does not want to watch the channel anymore, he or she can switch back to P-mode. The last recommended four channels are recommended again. Finally, when the user logs out of our platform, it records all of the information in the user logs and delivers it to the feedback module to train the P-tree.
and adjust the priority of each channel. Next time the user logs in, he or she can preview favorite channels quickly.

### 3.2.1. Preview database

There are two information was used in the Preview Database: (1) User Logs and (2) Channel information (see Table 1).

We use the information of $R_v$ to create the P-tree. Figure 10 shows the architecture of the P-tree of each user. Channels are classified into $M$ types, and each type includes $N$ related channels. We can express $P_{tree} = \{P_{i,j}|0 < i \leq M, 0 \leq j \leq N\}$ and sort them by the $R_v$. Therefore, when a user logs in, we follow the P-tree to recommend channels and show them in P-mode. The algorithm of P-tree is shown in algorithm 1.

**Algorithm 1 Create P_tree.**

1. $C := \text{the total number of channels}$
2. $L := \text{the tree level of } P_{tree}$
3. $R_v[] := \text{the list of recommendation value}$
4. $C = M \times N$
5. $L = \log_2 M + 1$
6. $i \leftarrow 1$
7. for $i = 1$ to $C$
8. read the $R_v$ into $P_{tree}$
9. end for

**Theorem 1.** The worst-case Create P_tree (Algorithm 1) overhead is $O(n)$.

*Proof.* Assume that the $T_p(n)$ is the maximum number of steps that can be executed for the given parameters, and $N$ indicates the maximum number of channels that is limited by an integer: Then we can determine $T_p(n) = N + 6$. Hence, the overhead of the worst case in the proposed algorithm is $O(n)$. □

### 3.2.2. Recommendation module

The RM is designed based on the concept of the loser tree [24]. Each leaf of a tree represents a unique channel type in the RM. The two nodes within the same group are compared, and the node with a smaller $R_v$ is the loser. The following is an example of the loser tree.

First, if leaf nodes of a tree are null, an $R_v$ of a new channel is read into leaf nodes. The maximum $R_v$ is determined using the loser tree algorithm. Figure 11 shows the result after the first round. The channel $A_1$ is recommended in the first round because it has the maximum $R_v$.

Similarly, channels with the second and the third high $R_v$ are recommended through the same process. According to Theorem 1, the user can preview $P_n$ channels simultaneously. In this example, $P_n = 4$, which means that the user can simultaneously preview four channels. Therefore, the algorithm recommended four channels, and a page $Page_n$, which contains the four channels, was completed and delivered to the user (Fig. 12). The Algorithm 2 shows the operation of the RM.

**Theorem 2.** The worst-case Recommended module (Algorithm 2) overhead is $O(\log_2 n)$.

*Proof.* The $R_v$ of each channel type is sorted at the beginning of the algorithm. Radix sort can be applied to the result, such that the elements in $R_v$ are all small integers, and the time complexity is $O(n)$ to sort $n$ elements in the range $[0 \cdots n^d]$, where $d$ is a constant. The channel ID with the highest $R_v$ is selected into the leaf node of each channel type, and the time complexity is $O(1)$. Two sibling nodes at each level are compared one time in the
A Fast SVC-Based Channel-Recommendation System

Algorithm 2 Recommendation module.
1: While Preview Page \( \neq \) Full and P-tree \( \neq \) null do
2: for \( j = 1 \) to \( M \)
3: if leaf node[ \( j \) ] = null
4: read the highest \( R_v \) into leaf node[ \( j \) ] from the channel type \( j \)
5: end if
6: end for
7: /* compare two node between sibling of each level */
8: for \( k = L \) to \( 1 \)
9: if L.node < R.node
10: Parent.node \( \leftarrow \) L.node
11: * Grandparent.node \( \leftarrow \) R.node
12: else if
13: Parent.node \( \leftarrow \) R.node
14: * Grandparent.node \( \leftarrow \) L.node
15: else
16: compare the channel favorite value \( f \)
17: end if
18: end for
19: end while

Loser tree, and the time complexity is \( O(\log_2 n) \). One preview page includes a maximum of four channels in the C2P platform. Therefore, the overhead of the worst case in the proposed algorithm is \( 4 \times [O(n) \times O(1) + O(\log_2 n)] = O(\log_2 n) \).

3.2.3. Mark module
The MM is designed to improve the efficiency of the channel recommendation through the ‘Mark’ function to dynamically adjust the \( R_v \). The main purpose of this module is that when users preview numerous channels, they could be indecisive on which channel to watch or they may want to preview more channels. A user’s indecision means that the user might be interested in those channels of similar types.

Algorithm 3 Mark module.
1: \( M_{total} \leftarrow \) the total number of mark in the channel type \( M \)
2: identify the mark channel in which channel type \( M \)
3: increasing the \( R_v \) in the type \( M \)
4: for \( i = 1 \) to \( N \)
5: \( R_{u(new)} = R_{u(old)} + M_{total} \)
6: end for

Theorem 3. The worst-case MM (Algorithm 3) overhead is \( O(n) \).

Proof. Assume that the \( T_p(n) \) is the maximum number of steps that can be executed for the given parameters. The marked module maximum updates \( N \) recommendation values \( (R_v) \) at the same time in the same type; therefore, we can determine \( T_p(n) = N + 3 \). Hence, the overhead of the worst case in the proposed algorithm is \( O(n) \).

At this time, the user can mark any channel or choose to do nothing. If the user does not mark any channel and directly moves to the next page, the system recommends the next Page\(_{n+1}\) through the RM again (Fig. 14).
Through the RM and MM, the fast channel recommendation is achieved. Therefore, the system adjusts all the channels and the number of total channels watched after the user marks channels. In this example, because two channels are marked, the system assumes that the user has a great opportunity to watch channels of the same type. However, if the user marks channels \( C_1 \) and \( C_2 \) in \( \text{Page}_n \), the system updates the channel recommendation system. Thus, we propose Formulas (3) and (4) to modify the \( R_v \):

\[
\begin{align*}
\text{New} & = \text{Old} + (\text{Total number of mark in this channel type}), \\
\text{New} & = \text{Old} + (\text{Total number of mark in this channel type}) \times \text{constant } \pi.
\end{align*}
\]

If a channel is only marked without being watched, its \( R_v \) is calculated according to Formula (3). If a channel is both marked and watched, we multiply its mark value by the constant \( \pi \) to enhance the feedback effect and the \( R_v \) is calculated using Formula (4).

The system is trained by the user’s historical log; thus, it is able to derive the channel preferences of each user, and enables users to quickly find their favorite channels without having to browse through numerous pages. In addition, this means that the frequent transmission of huge traffic of high-quality videos is unnecessary. The next section presents more details.

### 3.2.4. Feedback module

Through the RM and MM, the fast channel recommendation based on user preference is achieved. Moreover, the system records all user behavior, such as the user’s favorite channel, number of marked channels, marked channel types, watched channels and the number of total channels watched after the user logs out of our platform. This information is useful for the channel recommendation system. Thus, we propose Formulas (3) and (4) to modify the \( R_v \):

\[
\begin{align*}
R_v^{\text{new}} & = R_v^{\text{old}} + (\text{Total number of mark in this channel type}), \\
R_v^{\text{new}} & = R_v^{\text{old}} + (\text{Total number of mark in this channel type}) \times \text{constant } \pi.
\end{align*}
\]

If a channel is only marked without being watched, its \( R_v \) is calculated according to Formula (3). If a channel is both marked and watched, we multiply its mark value by the constant \( \pi \) to enhance the feedback effect and the \( R_v \) is calculated using Formula (4).

The system is trained by the user’s historical log; thus, it is able to derive the channel preferences of each user, and enables users to quickly find their favorite channels without having to browse through numerous pages. In addition, this means that the frequent transmission of huge traffic of high-quality videos is unnecessary. The next section presents more details.

### 4. SIMULATION RESULTS

In this section, we investigate the performance of the C2P platform, focusing on its predictive accuracy, feasibility and efficiency.

#### 4.1. The predictive accuracy analysis of FLT mechanism

The quality of a recommender system can be evaluated by predictive accuracy metrics, where the predicted ratings are directly compared with actual user ratings. The most commonly used metric in the literature is the mean absolute error (MAE) [25], which is defined as the average absolute difference between the predicted ratings and actual ratings:

\[
\text{MAE} = \frac{1}{N} \sum_{u,i} |p_{u,i} - r_{u,i}|,
\]

where \( p_{u,i} \) is the predicted rating for user \( u \) on item \( i \), \( r_{u,i} \) is the actual rating and \( N \) is the total number of ratings in the test set.

In the predictive accuracy experiment, we used the MovieLens [26] data sets, which consist of movie-ratings data collected using a web-based research recommendation system. The data sets contain 943 users, 1682 movie items and ~100 000 ratings on a scale from 1 to 5. The data sets are divided into training data sets (\( u_x \), base) and test data sets (\( u_x \), test) in an 80/20 ratio.

To validate whether the proposed platform architecture is feasible, we used MyEvalvid-NT [27] and myEvalSVC [28]
to simulate and evaluate multimedia video transmission. We designed the following environment. The ‘Foreman’ test file, which has a format of CIF(352*288) with 400 frames, was used; thereafter, the JSVM [29] was used for encoding and decoding. The experiment was divided into two simulated platforms. Table 2 presents the experimental parameters.

1. Pure P2P networks: Video files were sent directly to P2P networks. According to [30], the PLR (Packet Loss Rate) of P2P networks is ~0.005–0.02. Therefore, the PLR of Platform 1 in the experiment was set to 0.005–0.02.

2. Hybrid Cloud and P2P networks: The base layer video stream (Layer_0) was transmitted to the cloud networks, and the enhancement layer video streams (Layer_1 and Layer_2) were transmitted to the P2P networks. A test file named IPTV.scr (CISCO IP/TV, MPEG video stream) was used as the parameter to test the packet loss rate from Amazon EC2, and the average test result of the packet loss rate was 0.005. Therefore, the PLR of Platform 2 in the experiment was set to 0.005.

From this experimental environment, we produced RTF and STF files, in addition to the NALU filter to filter out the original NALU to trace the file to generate the filtered NALU trace file, through BitStreamExtractor to convert the filtered NALU trace file into H.264 format, and then filtered the YUV video decoded using the JSVM decoder. Finally, the Peak Signal-to-Noise Ratio (PSNR) tool was used to analyze the YUV video files after reconstruction. The results are shown in Figure 18.

Figure 19 shows that the PSNR values of the two experimental environments are close. The main causation of the quality of PSNR is the frame loss rate, where the higher frame loss rate causing low PSNR means poor video quality. A frame sent through the P2P networks has more of a chance to be lost, resulting in more unstable quality, and receiving lower average PSNR values. However, in the cloud and P2P hybrid networks, because the base layer frames were sent through the cloud network, the probability of frame loss was significantly lowered. With the lower overall frame loss rate, the PSNR value is relatively higher than that of the pure P2P networks. Therefore, the experimental results prove that the video quality delivered via our proposed platform can be accepted by the user, and is
Even of a higher quality than the original P2P networks, and also indirectly proves the feasibility of our proposed platform. Table 3 presents the average PSNR comparison results of the two platforms.

### Table 3. The average PSNR comparison results of the two platforms. 

<table>
<thead>
<tr>
<th>Platform</th>
<th>Average PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2P (Platform 1)</td>
<td>31.879058</td>
</tr>
<tr>
<td>P2P+Cloud (Platform 2)</td>
<td>34.574753</td>
</tr>
</tbody>
</table>

Finally, not only can the PSNR value of the reconstructed video be calculated using the program, but also the video content can be watched using the YUV viewer, as shown in Fig. 19.

In Fig. 19, the video on the left-hand side was sent to the P2P networks, and the video on the right-hand side was sent to the cloud and P2P hybrid networks. Both of the reconstructed videos were obtained at the receiver. We only took one frame for our comparison. Figure 18 obviously shows that the video quality on the right-hand side is better than that on the left-hand side. The experimental results of the PSNR match the results of viewing the video content, indicating the feasibility of the proposed platform again.

### 4.3. The efficiency of the C2P Platform

This section details the simulations we performed for analysis, and we compared the performance between the C2P platform and the HPMP system. In the HPMP system, the HTC is defined as the average channel-hopping time (ACHT), and the TFS algorithm is used to compute it. In our simulation, we transform the HTC as the switch page (SP) and compute it using the FLT algorithm. However, these two concepts are similar because the measurements both start from the client’s first channel to his or her favorite channel. A smaller value represents a shorter time users take to find their favorite channels.

To compare with HPMP system, we set the simulation parameters as follows: Twenty channels are classified into four channel types, and each type includes five channels. We assume that the favorite channel of each peer is selected from these channels. In total, 200 peers join our platform in 300 seconds. We set the period $T$ of auto-switching channel preview as 8 s.

Our first experiment involves determining which system could satisfy faster the request of all peers. We assume that each peer has only one favorite channel from Channels 1–20. Figure 20 shows the cumulative number of peers who have found their favorite channel. The peers in the C2P platform evidently are faster than those in the HPMP system.

Our FLT algorithm has the P-tree to record the favorite channel of each peer and to guarantee that each channel can
be previewed. When peers enter the preview mode, they can find their favorite channels in 40 s. This means that peers only have to switch between five pages to preview all 20 channels. Conversely, the HPMP system must rely on the most stable source peer on the watch mode. This causes a major problem that if some user wants to watch non-popular channels, they could spend a long time switching between pages to find a channel. Because non-popular channels cannot obtain a higher-quality value \( Q \), these channels do not appear in the preview page in a short time.

The HPMP system highly relies on the quality of the source peer. However, numerous channels are in a real IPTV environment. An insufficient number of source peers results in a low \( Q \) value, causing the HPMP system being difficult to recommend non-popular channels. Figure 21 shows that the performance of the proposed system is superior to that of the HPMP system. To make rigid experimental results, we increase our total number of channels to 80 within eight channel types. (Each type includes 10 channels.) We consider a new condition that the users change their favorite channels when they lose interest in old channels.

The value of SP is shown to have a convergence trend in our C2P system. Even if peers change their favorite channel, the SP value remains in convergence. With FLT, the P-tree updates the channels according to a peer’s operating logs, and the system dynamically adjusts the \( R_v \) of each channel to ensure that the favorite channel progressively rises to the top. If the favorite channel moves into the top four of the P-tree, the peer finds his or her favorite channel in the first page after logging into our system. Conversely, the HPMP system retains random probability to show the favorite channel.

Finally, we conducted a third experiment to prove that the FLT algorithm is able to sort favorite channels. After training with enough user logs, the favorite channels move to the front page. Figure 22 shows the number of pages that users can be satisfied. Most users in C2P can clearly find their favorite channel in two to five pages. Conversely, the HPMP system has a lower degree of satisfaction because of the fact that its channel recommendation method relies on quality.

5. CONCLUSION

This study proposed a fast SVC-based channel-recommendation system for IPTV on a Cloud and P2P hybrid platform. The system provides a fast channel-switching mechanism hybrid Cloud and P2P networks. Experimental results show the feasibility of our proposed platform by indicating that the proposed platform can obtain a higher PSNR quality compared with the original P2P networks. The higher PSNR indicates that the proposed system occurs video freeze less frequently than HPMP. We propose a novel recommendation system based on the FLT algorithm. The proposed system can help users find their favorite channels in only 2–5 switching pages. The performance of the proposed system is \( \sim 75\% \) better than that of the HPMP system. In the future, we will work toward optimizing the recommendation content of the preview channels.

FUNDING

This work was supported by National Science Council (NSC) project of Taiwan (NSC-102-2221-E-415-012-), and the Information and Communications Research Laboratories (ICL), Industrial Technology Research Institute (ITRI), Taiwan, Republic of China. Furthermore, we wish to thank Zih-Huan Hong for his assistance in collecting the experimental data.
REFERENCES