In a Peer-to-Peer (P2P) environment, which can improve scalability and lessen bandwidth waste, the load of the Video-on-demand (VoD) services can be distributed to reduce the burden on centralized video servers and maximize the bandwidth utilization. Conventional P2P techniques for VoD services only consider data between active peers in the same VoD session. The inactive peers that leave the session but still hold some of the media content in their local storage are ignored. In this article, we design and implement a fully decentralized VoD service via P2P techniques, which is referred to as the MegaDrop system. The MegaDrop system not only takes active peers into consideration but also provides mechanisms for discovering inactive peers that contain desired media objects. Evaluation results of the MegaDrop system show that our architecture performs more efficiently when the more inactive peers involving to provide media blocks.

Keywords: peer-to-peer, video-on-demand, cooperative VoD, multilayer
their participation in some VoD session: (1) active peers: are currently joining at least one VoD session and can share some of the media content in their playback buffer so that these media clips can be shared to other peers with a stable availability, (2) inactive peers: have left the session but stays online, where the media contents that has already played out but still remained in their local storage and can be fully or partially shared to other peers, and (3) left peers: have left the P2P network so that the media clips they ever hold earlier are no longer accessible to other peers unless they rejoin the P2P network. In the past, various approaches to VoD streaming using P2P technologies, such as Chaining [14], DirectStream [21], P2VoD [17], P2Cast [22], a proximity-aware strategy [9] and network coding [4] for multi-overlay P2P streaming, had little attention about considering reusing the media resources that inactive peers hold in their local storages. Streaming the remaining video content in a VoD session involving the inactive users not only provides more potential resources for the peers but also increases the efficiency of VoD service.

In this paper, we aim to design a new architecture supporting VoD streaming services over a P2P environment. The proposed architecture, referred to as MegaDrop, is based on a multioverlay network. The first overlay network is used to find a specific object, and the second overlay network is used to stream media data. The architecture is fully decentralized without any central administrative points. To define the service boundary at different levels, we further partition the architecture into four tiers: Peer Discovery Layer (PDL), Content Lookup Layer (CLL), Media Streaming Layer (MSL), and Playback Control Layer (PCL). The PDL is responsible for finding peers and constructing the first overlay network. The CLL generates a unique identifier for a media object and provides a content-matching capability. The MSL transmits media between peers and constructs the second overlay network. The PCL interacts with end users by providing the interface of control operations.

The MegaDrop system contributes several advantages. First, the architecture is fully decentralized without any central administrative points that suffer the single point of failure. Second, the promising multioverlay construction across the multilayer architecture precisely decomposes the primitive technologies, forming a P2P VoD streaming, into each layer so that each layer maintains the flexibility and extensibility without interference. Likewise it represents that each layer can be replaced individually and painless by newly invented technologies or customizations. Third, the inactive peers who ever played the media contents can assist other peers in playing the same media contents hereafter. By the concept of mutual assistance in the P2P system, the efficiency and utilization of network bandwidth could be leveraged as expected.

The remainder of the paper is organized as follows. Section 2 provides related works about the VoD streaming in P2P environments for reference. Section 3 introduces an overview of the proposed MegaDrop system. Section 4 provides the detailed and implementation of the system components in the architecture. Section 5 presents the performance evaluation of the MegaDrop system from several essential metrics. Finally, concluding remarks are made in section 6.

2. RELATED WORKS

To the best of our knowledge, the Chaining technique proposed by Sheu et al. [14]
firstly applied the P2P concept to the VoD streaming. Each client in Chaining has a fixed-size buffer to cache the most recent content of the video stream that it has received. A new client can receive the video stream from an early client as long as the first data block of the video is still in the buffer of the latter. However, this scheme does not provide a recovery protocol in case of failures, thus clients quitting a chain cause the chain to be broken. Souza et al. [10] proposed a virtual server mechanism to establish and host a group of peers so that clients can aggregate necessary resources via sending requests. The overhead of control requirements in the virtual server is also considered. Compare to Chaining [14], the analytical result show that DynaPeer [10] improves VoD system capacity and decreases the server-load. In [21], DirectStream constructs tree structures to deliver video based on an interval-caching scheme similar to Chaining. DirectStream consists of a directory server, content servers, and clients. The directory server works as a central administrative point, which is analogous to the Napster server [15]. The directory keeps track of the information of all servers and clients, assisting new clients to join a session and locate the required service. The clients in DirectStream cache a moving window of video to serve other clients like a server by forwarding the stream. One drawback in DirectStream is the deployment of the directory server. The directory server may suffer a single point of failure.

P2Cast [22] is derived from the traditional patching technique with unicast connections among peers. P2Cast clients not only receive the requested stream, but also contribute to the overall VoD service by forwarding the stream to other clients. One limitation in P2Cast is that clients only cache the initial part of the video. Hence, two streams (a base and a patching stream) are used at the beginning to deliver the video to a late client. In addition, P2Cast has to get the source involved whenever a failure occurs. Thus, it is vulnerable to disruption due to server bottlenecks at the source.

P2VoD [17], similar to P2Cast, makes a late client obtain the initial missing part of a video (namely, a patch) not only from the server but also from other clients. Clients in P2VoD always cache the most recent content of the video stream, while clients in P2Cast only cache the initial part of the video. P2VoD handles failures locally and without involvement of the source, thereby reducing the blocking time before orphaned peers can resume. The major disadvantage of P2VoD is a possible long recovery time due to a client having an out-of-date list of its siblings’ IP addresses.

Besides, more and more P2P streaming mechanisms are proposed to improve large-scale multimedia distribution, such as a proximity-aware mechanism [9] in P2P streaming can reduce delay and load on network by removing long-haul unicast connections, in tandem with development for a more complicated but real application. A network coding based media distribution strategy [4] was proposed to accommodate multiple coexisting streaming overlays for a P2P live streaming IPTV application. SonicVoD in [11] also exploits a push-based mesh streaming scheme based on network coding to distribute the video to increase system resilience. In [23], VOVO aims to shorten the latency for playback interactivity with a P2P streaming VCR by prefetching the requested video content through a state propagation gossip protocol among peers. However, none of them considers the media resources held by the inactive peers. With the contribution from the inactive peers, the collaboration efficiency in a P2P environment can be significantly raised.
3. SYSTEM ARCHITECTURE

3.1 Architecture of the MegaDrop System

The system architecture shown in Fig. 1 illustrates the VoD functions, such as locating, streaming, and controlling and playing back a media object, in the Megadrop system over a P2P network. The four layers – PDL, CLL, MSL, and PCL in the Megadrop system are described in details in the following subsections.

![Multilayer architecture of the MegaDrop system.](image1)

3.1.1 Peer discovery layer

The PDL provides an efficient peer lookup service such that upper layers can use to search for desired media objects propitiously. It contains several major functions: routing messages between peers, filtering out unnecessary messages, caching queries within peers, and maintaining peer information. A notable feature is that the PDL is not restricted to a specific P2P network, and hence any typical P2P overlay network such as Gnutella and Pastry [1], Chord [6], or CAN [13] can be adopted and wrapped in this layer. As shown in Fig. 2, the PDL categorizes peers into two types: trunks and branches to minimize...
searching and routing traffic while reserving the bandwidth for media transmission. Trunks are responsible for forming the overlay backbone, organizing surrounding nodes, filtering and directing traffic over several message types. The nodes with more powerful capability are selected to be trunk nodes. In contrast, a node with limited resources, such as, limited bandwidth, low processing capability is treated as a branch node. This division allows a peer to find a media title easier.

3.1.2 Content lookup layer

The CLL acts as the mediator above the PDL. At least two basic operations should be handled in this layer: (1) when a peer is willing to share a media object, a corresponding media-info, which is defined in section 3.2.2, has to be generated by the CLL (e.g., the CLL keeps a media pool for maintaining the media-info structures of shared media objects) and (2) the CLL should aid the content-aware search queries received by the PDL, so as to allow the PDL to focus on routing messages/queries and ignore processing of the media content. A mechanism for error recovery to ensure the completeness of received media blocks could also optionally be implemented in this layer.

3.1.3 Media streaming layer

Media transmissions between peers are handled by the MSL. The MSL does not know which peers hold the media objects, and must rely on the cooperation from the CLL and the PDL to find the desired objects. The peers in the MegaDrop system can be classified into three types: suppliers, brokers, and droppers. A supplier is a peer that has a complete media object, whereas a dropper has an incomplete media object and hence needs to retrieve missing media blocks from other peers. Therefore, a supplier is always capable of providing any portions of a media object, while a dropper can only provide an incomplete media object. Furthermore, a broker is an intermediate peer that maintains peers lists to assist the exchange of peer information between suppliers and droppers via the field of brokers in media-info.

![Media streaming layer](image-url)
As Fig. 3 shows, during a media session, several peers form a cluster virtually, where some peers act as suppliers and some act as droppers. Among these peers, one peer would be treated as a broker, which is connected with other peers and maintains the list of sending peers. For a virtual cluster, there should exist at least one supplier or several droppers containing exclusive media blocks to restore an entire media object.

In addition, to minimize the unfairness or freeriding that commonly exists in P2P environments due to unequal contributions by peers, it is desirable to implement a bartering technique to provide each peer with incentives to contribute media objects.

### 3.1.4 Playback control layer

The PCL is designed to directly interact with end users and provides a user interface for video playback and administrating via VCR-like operations such as playing, pausing, stopping, seeking, and fast forwarding. The PCL cooperates with the MSL to change the caching policy (via buffer management) depending on the behavior of users. For example, when a user issues a seek operation to change the current play point, the PCL needs to notify the MSL to acquire the media blocks starting at the new play point first. Moreover, for convenience in controlling the playback, the PCL should provide the status of the monitoring and operational statistics for users to manage the system performance.

### 3.2 Media Representations

According to the aforementioned characteristics of the MegaDrop system, we devise a media representation that breaks the content of a typical media object into media blocks based on a group of picture. It allows both active and inactive peers to potentially share their media content to form media streaming. Then each peer that only holds some of the media data can gather the remaining required media data from multiple peers. The proposed media representation provides several advantages: (1) the loss of some of the media blocks during media transmission would not damage the media object or make it undecodable; (2) the amount of data involved in the error recovery or media retransmission in some reliable conditions can be reduced to the unit of a media block; and (3) it is easy to implement some VCR-like operations (e.g., fast forward, fast rewind, and jump forward/backward) at multiples of the normal playback speed by skipping media data on the basis of multiple media blocks.

A single media object may be spread over multiple peers, and hence the receiving peer must know how to gather multiple portions of the media blocks from other peers. We therefore introduce an original structure, called *media-info*, to provide a unique identifier that helps a receiving peer locate and restore a specific media object. Basically, *media-info* provides the global information of a media object, such as the frame rate, video codec, media title, creation time, author information, and copyright information, and is followed by hashed information. In the following subsections, we first describe the hashing procedure to generate the hashed information, called *media-hash*, and then use this to construct the original *media-info*. 
3.2.1 Media-hash

A media-hash is the unique identifier of a media object. As depicted in Fig. 4, media-hash is produced by a two-step hash scheme. First, a specific hash function $H(x)$ (e.g., Secure Hash Algorithm 1, cyclic redundancy check 32, or Message-Digest Algorithm 5) is applied to each media block to produce a hashed block. Then, media size, hash scheme, and hash size together with the concatenated hashed blocks are hashed again by $H(x)$ to generate the media-hash, which uniquely represents a media object. $B_n$ denotes the $n$th media block, where $1 \leq n \leq N$. $B_{len}$ represents the length of a media block, which generally would be chosen as 128, 256, 512, or 1024 KB. Thus $B_n$ can be computed as $\left\lceil \frac{M_{len}}{B_{len}} \right\rceil$, where $M_{len}$ is the size of the media object (in bytes) and $\lceil \rceil$ denotes the ceiling operation. Furthermore, $HB_n$ and $H_{len}$ (in bytes) indicate the value and length of the hashed data for media block $B_n$, respectively (i.e., $HB_n = H(B_n)$ and $H_{len} = \text{length of } HB_n$). Zero values would be padded to the last media block if its size is less than $B_{len}$. Before applying the second step of hashing, the size of the hashed data can be calculated as $H_{len} \times B_N$, the media size equals to $M_{len}$, and the field of hash scheme merely specifies what kind of hash function is used during this procedure. In particular, the size of the produced media-hash equals to $H_{len}$ if only one hash function is chosen for the two-step hash scheme procedure. Through this procedure, the media-hash is constructed for the purpose of uniquely identifying a media object. By constructing media-hash, we can compute the ratio of the overhead size to the media size and assist in deciding a proper size of a media block for a specific network bandwidth during a media transmission, which will be explained later.

3.2.2 Media-info

Media-info is created to provide not only the unique identifier for a specific media object through media-hash, but also sufficient information for a receiving peer to know (1) where the media object is stored; (2) how to gather and stream the media object; and (3) the characteristics of a media object sufficient for decoding and rendering it. Conse-
JIA-MING CHEN, JENQ-SHIOU LEU, YEN-CHIU CHEN, HSIN-WEN WEI AND WEI-KUAN SHIH

Fig. 5. The structure of media-info.

Fig. 5. The structure of media-info.

sequently, media-info should be retrieved before a streaming session starts. Fig. 5 shows the structure of media-info. The fields of media title, date time, creator, and comments are extracted from the original header of the media object, which are optionally provided by media-info. Note that media-info also contains hashed blocks, which could be used to verify the correctness of each retrieved media block for the purpose of error recovery. Additionally, the field of brokers tells the receiving peer how to gather and stream the media blocks, which are probably spread among multiple peers. A media-info is generated in section 4.2 and the detailed usage of this field is described in section 4.3.

The size of media-info, I_len, is highly dependent on the size chosen for each media block:

\[ I_{len} = \left( \frac{M_{len}}{B_{len}} \right) \times H_{len} + NF_{len} + OF_{len} \]  

(1)

where \( M_{len} \), \( B_{len} \), and \( H_{len} \) are defined as in section 3.2.1, and \( NF_{len} \) and \( OF_{len} \) represent the sizes of the mandatory fields (including the brokers’ field) and media header fields in Fig. 5 respectively. A smaller media-info leads to a larger \( B_{len} \). However, a larger \( B_{len} \) produces a smaller number of media blocks, which potentially reduces the degree of concurrency for media transmission among multiple peers.

4. IMPLEMENTATION OF THE MEGADROP SYSTEM

In this section, we present the comprehensive implementation of crucial operations in each layer to realize the MegaDrop system.

4.1 Peer Discovery Layer

The PDL is implemented by modifying an LGPL library, GnuDNA [26], which is based on the Gnutella2 [25] overlay network. In our experience the topology shown in Fig. 2 minimizes the searching and routing traffic, thereby reserving network bandwidth for media transmission. Theoretically, an adequate number of trunks (or peers) allows a peer to find desired media blocks in the network.

The following major operations are implemented in the PDL. First, bootstrapping allows a peer to participate in the MegaDrop system. We utilized GWebCache [27] to allow a peer to discover other peers which are currently available for connection. Second,
after obtaining several IP addresses via a bootstrapping procedure, a peer becomes a
trunk or branch by handshaking with these IP addresses. We adopt a common three-way
handshaking mechanism to develop this protocol. Third, the communications between
peers are unified by replicating a subset of the messaging system in the Gnutella2 proto-
col. Likewise, query hash tables are utilized to facilitate the query-routing protocol,
which is implemented by modifying Prinkey’s scheme [28].

4.2 Content Lookup Layer

Any behaviors reliant on the media content could be implemented in the CLL. Here
we have implemented three major operations. First, to generate media-info, we use the
Secure Hash Algorithm 1 (SHA1, an algorithm for computing a condensed representa-
tion of a message) as our hash scheme, which generates each hashed block in 20 Bytes.
Second, the CLL implements content searching/matching based on the media content
rather than keyword searching by the PDL. Providing content searches in advance can
complement keyword searches because media objects are usually not uniquely identified
by keywords. Third, data integrity is ensured by hashing and comparing a received me-
dia according to the information provided by media-info. Thereafter, the CLL can coop-
erate with the MSL to decide whether to drop or retransmit a received media block that is
corrupted.

4.3 Media Streaming Layer

Within a media session, the MSL performs media transmission across multiple
sending peers. We employ a variant of BitTorrent protocol [2] to accomplish most of the
functions and several operations in this layer. First, media-info is requested since the
MSL needs this information before establishing a media session. Second, due to the ver-
satile status of active peers in a P2P environment, a receiving peer can periodically up-
date the list of sending peers from the broker through brokering mechanism (note that a
broker maintains a global view in order to allow a media object to be aggregated). This
protocol is realized simply using HTTP conventions. Third, media streaming enables a
receiving peer to maintain TCP/IP connections with sending peers. In our implementation,
this procedure contains three sub-functions: (1) handshaking, which ensures a TCP/IP
connection to be established before a sending peer serves the desired media object; (2)
connection-state guarding, which tracks the states of remote peers (as three modes:
choking, un-choking, or interesting) in monitoring the connection so that requests and
media data can be transferred only when states of both peers are un-choking or interesting;
and (3) control/media data communication, which transfers control or media data between
two peers. Fourth, to deliver media blocks efficiently in the MSL, every media block can
be further decomposed into several smaller units called media pieces, such that a single
media piece is treated as the smallest transmission unit between peers. The buffer space
available at sending and receiving peers can act as a cache to manage these media pieces.
In our system, some specific download policies are implemented to speed up the trans-
mission between peers, such as an unpredictable (random) order, the rarest piece first,
and so on. Therefore, the media pieces may not be delivered in the correct order in our
system. Finally, a variant of the tit-for-tat scheme is used to implement bartering. Fig. 6
demonstrates the temporal access flow of a media session in the MSL, in which requesting media-info and handshaking are executed once, whereas brokering is periodically executed and message communication is frequently executed.

### 4.4 Playback Control Layer

We have implemented the PCL via the Microsoft™ Developer Network (MSDN) [24]. DirectShow is essentially built on a group of filters, each of which performs a specific operation for streaming a media file. Connecting several filters by different input/output ports forms a filter graph. In this implementation, we additionally introduce a high-level component, called the Filter Graph Manager, to control the flow of a filter graph. Fig. 7 displays the filter graph based playback interface of MegaDrop, where a source filter directs the media data that may come from local storage, the network, or capture devices to the transform filter, which could be decoders, encoders, splitters, or multiplexers. Then, the media data are output to display devices via rendering filters. Such an architecture can make the MegaDrop system suitable for a variety of codecs (e.g., MPEG series, H.264, WMV series), thereby increasing its flexibility.

Instead of implementing these filters from scratch, our implementation is based on filters provided by Microsoft™ SDK. Actually, we design a customized source filter to collaborate with the MSL, such that buffer management with a request-on-demand policy
could be seamlessly connected to the source filter. A transform filter is added to aid the MSL in gathering or splitting the media pieces. Finally, the overall VCR-like operations have been carried out through the IMediaControl interface in DirectShow, which can indicate the Filter Graph Manager to control the media flow. Our implementation of PCL not only supports playback operations within the MegaDrop system, but also could be executed as a purely local media player.

5. PERFORMANCE EVALUATION

In this section we primarily focus on evaluating the performance of the MegaDrop system from the following essential metrics: (1) the startup delay versus the size of the media-info before starting a media transmission; (2) the efficiency of bandwidth utilizations by measuring the overheads induced from network traffics other than real transmitted media blocks; and (3) the effectiveness of bandwidth aggregations among multiple peers; (4) the effectiveness of the total streaming time in terms of different percentages of inactive peers involved in MegaDrop compared to the existing solution without inactive peers involved. The detail discussions are summarized in the following subsections.

5.1 Startup Delay

In addition to the play out policy on the performance of P2P streaming system [3], a critical issue for a VoD streaming system is whether it can deliver a video stream to users without excessive startup delays which affect the playback quality experienced by users. Therefore the startup delay is evaluated in most of P2P streaming systems [8]. In the MegaDrop, the startup delay is significantly affected by the size of the media-info since a media-info must be delivered to the requesting peer before media transmission. We assume that the maximum startup delay for a media session is $D$ (in seconds) and the average network speed is $BS$ (in bytes/second). Based on Eq. (1) listed in section 3.2.2 we can derive the minimal size of the media block, $B_{len}$, required to satisfy the maximum tolerable startup delay $D$:

$$B_{len} \geq \frac{M_{len} \times H_{len}}{BS \times D - NF_{len} - OF_{len}}.$$

Fig. 8 illustrates the relationship between the number of media blocks, the size of media-info, and media size for various media objects. The derived results reveal that the length of media block should be carefully chosen for having a suitable size of the media-info to fulfill the tolerable startup delay in the MegaDrop system.

5.2 Efficiency of Bandwidth Utilization

The transmission of data in the system is treated as traffic overhead, such as that for media-info structures, messages for brokering, and other control messages within the PDL. We can use this criterion to evaluate the efficiency of the MegaDrop system. The results shown in Fig. 9 indicate that the overhead ratio arises significantly only when the...
media object is smaller than 10 kB. Even for the worst overhead ratio, 2.4% is still very low. Given that most of media objects are significantly larger than 10 kB, the MegaDrop system exhibits high bandwidth utilization.

5.3 Effectiveness of Bandwidth Aggregation

To investigate the effectiveness of bandwidth aggregating among peers, we setup an environment with four PCs in different specifications as shown in Table 1. In this simulation, there are five video clips with various sizes, ranging from 200 MB to 1000 MB, and the length of each media block is fixed at 256 kB for each video clips. During the simulation, up to 16 peers are constructed to join a media session by selecting PCs in sequence order of Table 1. That is, if a media session contains 11 peers, then PCA6, PCA1, and PCP4 acts as three peers, individually while NBP3 only acts as two peers. We select PCA6 as the broker in all scenarios due to its superior equipments. Besides, to approach
Table 1. Specifications of the simulation environment.

<table>
<thead>
<tr>
<th>Name</th>
<th>PCA6</th>
<th>PCA1</th>
<th>PCP4</th>
<th>NBP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>Desktop PC</td>
<td>Desktop PC</td>
<td>Desktop PC</td>
<td>Notebook</td>
</tr>
<tr>
<td>CPU</td>
<td>AMD Athlon XP 1.6 GHz</td>
<td>AMD Athlon XP 1 GHz</td>
<td>Intel P4 1.5 GHz</td>
<td>Intel P3 Mobile 1.06 GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>DDR 512 MB</td>
<td>DDR 256 MB</td>
<td>DDR 384 MB</td>
<td>SD 256 MB</td>
</tr>
<tr>
<td>OS</td>
<td>Microsoft Windows XP Professional SP2</td>
<td>Microsoft Windows XP Professional SP1</td>
<td>Microsoft Windows 2000 Workstation SP4</td>
<td>Microsoft Windows XP Home SP2</td>
</tr>
<tr>
<td>Network</td>
<td>Ethernet 100 Mb</td>
<td>Ethernet 100 Mb</td>
<td>Ethernet 100 Mb</td>
<td>Ethernet 100 Mb</td>
</tr>
</tbody>
</table>

Fig. 10. Average transmission time vs. number of participating peers in different media size.

the reality of the scenario, we have limited the total uplink bandwidth to 1000 kB and uplink bandwidth to 200 kB per media connection for each peer, such that bartering mechanism would be triggered while bandwidth is running out. Therefore, during a media session, the join time of a peer, denoted as \( T_{Ji} \), can be determined by Eq. (3), where \( N \) is the total number of peers in each test case.

\[
T_{Ji} = \begin{cases} 
0, & \text{when } i = 1, 2 \\
\frac{Mlen \times (i - 2)}{200 \times 1024 \times (N - 1)}, & \text{when } 3 \leq i \leq N.
\end{cases}
\]  

(3)

Fig. 10 shows the results of the simulation. As expected, the average transmission time per media session is reduced when more peers participate since this enlarges the degree of bandwidth aggregation. Especially for a large media object, the descending slope is more significant but is kept at a certain constant level, indicating that the effect of bandwidth aggregation is limited by the uplink bandwidth setting in the simulation.

5.4 Effectiveness of Total Streaming Time

Next, we demonstrate the effectiveness of the total streaming time in terms of different percentages of inactive peers involved in the MegaDrop system compared to the exist-
Fig. 11. Success rate of discovering media block among peers vs. different amounts of peers when different percent of all inactive peers are involved.

Fig. 12. Total streaming time vs. different block numbers when different percent of all inactive peers are involved.

According that a P2P network topology follows the power-law rule, the simulation model is generated by referring to a random graph model [18]. The total number of peers in the network varies from 10 to 1000, of which different percentages in active peers, inactive peers, and left peers are arranged to simulate the dynamic behaviors of peers in the P2P environment. The result in Fig. 11 shows that the higher percentage of the inactive peers leads to increase the successful ratio of discovering the media blocks under the same amount of peers in the MegaDrop system. On the other hand, the result in Fig. 12 shows the total streaming time in MegaDrop is shorter than the existing ones without the assistance from the inactive peers and also exhibits that the higher percentage of the inactive peers can reduce the streaming time needed in the MegaDrop system.

6. CONCLUSIONS

In this paper, we propose a practical design of MegaDrop to realize VoD services in
decentralized P2P environments. Compared to the existing mechanisms for VoD services, the MegaDrop architecture does not rely on any centralized server. Instead, every media object is delivered and propagated over the P2P network and every peer acting as a mini-server that retrieves and forwards media content from other peers. The processing and sharing of the same media object between multiple peers results in a formation of virtual server clusters. Meanwhile, the MegaDrop not only takes active peers into consideration but also provides efficient mechanisms for discovering inactive peers that contain desired media objects. The experiment results reveal that our approach is appropriate for serving VoD streaming, especially in the delivery of huge amounts of media data. Moreover, more participated peers in a media session can result in higher bandwidth aggregation, and reducing the average time required to stream a given media object. In addition, the workloads among peers can be balanced efficiently of and the network bandwidth can be well utilized. Conclusively, MegaDrop realizes an efficient and cooperative VoD system in P2P environments.

REFERENCES

11. L. Yu, L. Gao, J. Zhao, and X. Wang, “SonicVoD: A VCR-supported P2P-VoD


Jia-Ming Chen (陳家明) received the M.S. and the Ph.D. degrees in Computer Science from National Tsing Hua University, Taiwan, in 1998 and 2010 respectively. His current researches include power-aware, real-time and multimedia technologies for the heterogeneous multi-core SoC design, involving RTOS, Audio/Video codec, VLIW DSP (algorithm, programming, and architecture), multimedia framework, and P2P internet technology.
Jenq-Shiou Leu (呂政修) received his B.S. degree in Mathematics and his M.S. degree in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in 1991 and 1993, respectively. He was with Rising Star Technology, Taichung, Taiwan, as a R&D Engineer from 1995 to 1997, and worked in the telecommunication industry (Mobitai Communications and Taiwan Mobile) from 1997 to 2007. He obtained the Ph.D. degree on a part-time basis in Computer Science from National Tsing Hua University, HsingChu, Taiwan, in 2006. In Feb. 2007, he joined the Department of Electronic Engineering at National Taiwan University of Science and Technology, Taipei, Taiwan, as an Assistant Professor. He becomes an Associate Professor since Feb. 2011. His research interests include mobile services over heterogeneous networks, heterogeneous network integration and P2P networking.

Yen-Chiu Chen (陳燕菽) received her Ph.D. in Department of Computer Science, July, 2010 after she applied for a direct Ph.D. program from National Tsing Hua University in Taiwan in 2005. She received her B.S. degree in Department of Information Management from Chung Hua University in 2004. Dr. Chen is currently a postdoctoral researcher at the IC Design Technology Center, National Tsing Hua University in Taiwan. Her research interests include scheduling theory, real-time systems, embedded computer systems, and P2P network.

Hsin-Wen Wei (衛信文) received her Ph.D. degree in Computer Science from National Tsing Hua University, Hsingchu, Taiwan. Dr. Wei currently is a postdoctoral fellow in Institute of Information Science, Academia Sinica, Taipei, Taiwan. Her research interests include scheduling theory, real-time systems, and P2P networking.

Wei-Kuan Shih (石維寬) received his B.S. and M.S. degrees from National Taiwan University, Taipei, Taiwan and Ph.D. degree from University of Illinois, Urbana-Champaign. He is currently a full Professor at National Tsing Hua University, Hsingchu, Taiwan. His research focuses on real-time systems, wireless and personal communication system, internet technology, multimedia system.