DESIGN AND IMPLEMENTATION OF VIDEO ON DEMAND SERVICES OVER A PEER-TO-PEER MULTIOVERLAY NETWORK

Jia-Ming Chen, Jenq-Shiou Leu, Hsin-Wen Wei, Li-Ping Tung, Yen-Ting Chou, Wei-Kuan Shih
Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan

Keywords: Peer-to-Peer, Video-on-Demand, Overlay Network.

Abstract: Video-on-Demand (VoD) services using peer-to-peer (P2P) technologies benefit by balancing load among clients and maximizing their bandwidth utilization to reduce the burden on central video servers with the single point of failure. Conventional P2P techniques for realizing VoD services only consider data between active peers in the same VoD session. They never consider those inactive peers that have left the session but may still hold partial media content in their local storage. In this article, we propose a novel architecture to construct a fully decentralized P2P overlay network for VoD streaming services based on a multioverlay concept. The architecture is referred to as MegaDrop. It not only takes the types of peers into consideration but also provides mechanisms for discovering nodes that may contain desired media objects. Such a P2P-based scheme can distribute media among peers, allow peers to search for a specific media object over the entire network efficiently, and stream the media object from a group of the peers. We employ a layered architecture consisting of four major tiers: Peer Discovery Layer, Content Lookup Layer, Media Streaming Layer, and Playback Control Layer. The evaluation results show that our architecture is particularly efficient for huge media delivery and multiuser streaming sessions.

1 INTRODUCTION

Video-on-demand (VoD) services generally rely on one or only a small number of video servers to deliver video content. This topology places a heavy processing load on video servers, with the client–server paths representing a heavy network burden. Conventional techniques such as batching, patching (Kien, 2003), broadcasting, IP multicasting, proxy caching, and content distribution networks rely on centralized servers for delivering media. Peer-to-peer (P2P) computing represents another possible solution to such problems in media streaming.

The peers in P2P communications interact with others without the use of intermediaries, and can thereby reduce the burden on the servers delivering the media and increase bandwidth utilization. To the best of our knowledge, the Chaining technique (Simon, 1997) is the first to apply the P2P concept to VoD streaming. Each client in Chaining has a fixed-size buffer to cache the most recent content of the video stream it has received, but this scheme does not provide a recovery protocol in case of failures. DirectStream (Yang, 2003a) constructs tree structures to deliver video based on an interval-caching scheme similar to Chaining. However, the directory server in DirectStream represents a possible single point of failure. P2Cast (Yang, 2003b) is derived from the traditional patching technique with unicast connections among peers, but it is vulnerable to disruption due to server bottlenecks at the source. P2VoD (Tai, 2004) works similarly to P2Cast, and 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, and handles failures locally without the involvement of the source. 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.

The aforementioned schemes do not consider those inactive peers that have left the session but may still hold some of the media content in their local storage. Involving these inactive users in a VoD session for streaming the video content they still hold will provide peers with more potential resources in the same session, which could increase the effectiveness and efficiency of the VoD system. This scheme requires a mechanism for a requesting user to find video objects among both active and inactive users, which may be possible using generic P2P architectures with efficient lookup algorithms.

Based on the above, we aimed to design a new architecture supporting VoD streaming services. 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. Finally, we performed a series of experiments on the MegaDrop system to evaluate the startup delay, efficiency of bandwidth utilization, and effectiveness of peer bandwidth aggregation. The result shows that the aggregate bandwidth available to peers increases with the number of peers participating in a session, which thereby reduces the average streaming time.

The remainder of this paper is organized as follows. Section 2 introduces the media representations to properly fulfill the characteristics of a VoD streaming environment stated above. Section 3 provides an overview of the MegaDrop system, and the detailed design and implementation of the various components in the architecture are described in Section 4. Section 5 presents a performance evaluation of the MegaDrop system, and Section 6 proposes some ideas about how to consolidate this system. Finally, concluding remarks are made in Section 7.

2 MEDIA REPRESENTATIONS

According to the aforementioned characteristics, we devise a media representation that breaks the content of a typical media object (usually a media file) into media blocks based on a group of pictures (or frames). It allows both active and inactive peers to potentially share their (incomplete) media content to form media streaming, or each peer to only hold some of the media data and gather the remaining required media data from multiple peers. The proposed media representation provides several advantages: (i) the loss of some of the media blocks during media transmission would not damage the media object or make it undecodable, (ii) the amount of data involved in the error recovery or retransmission of the media data under certain conditions of unreliability can be reduced to the unit of a media block, and (iii) it is easy to implement certain VCR-like interactions (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 introduced an original structure, called *media-info*, to provide a unique identifier for a specific media object that allows it to be located and restored by a receiving peer. 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*.

2.1 Media-Hash

As depicted in Figure 1, *media-hash* is produced by a two-step hash scheme. First, a specific hash function $H(x)$ (e.g., SHA1, CRC32, or MD5) is applied to each media block to produce a hashed block. Then, those hashed blocks together with *media size*, *hash scheme*, and *hash size* are hashed again by $H(x)$ to generate the *media-hash*, which uniquely represents a media object. In Figure 1, B_n denotes the n-th media block, where $1 \leq n \leq B_N$. 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 $\lceil M_{len}/B_{len} \rceil$, 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} = length of HB_n). Note that zero values are padded to the last media block if its size is less than B_{len}. Obviously, before applying the second step of hashing, the hash size could be calculated as $H_{len} \times B_N$, the media size equals to M_{len}, and the field of hash scheme merely specifies the hash function used in this procedure. In particular, since we only choose one hash function in this two-step hash scheme procedure, the size of produced *media-hash* also equals H_{len}.

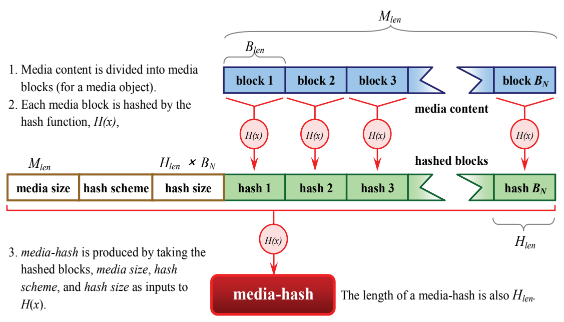
M_{len}
B_{len}
1. Media content is divided into media blocks (for a media object).
2. Each media block is hashed by the hash function, *H(x)*,
block 1
block 2
block 3
block B_N
media content
H(x)
hashed blocks
M_{len}
H_{len} × B_N
media size
hash scheme
hash size
hash 1
hash 2
hash 3
hash B_N
H_{len}
3. *media-hash* is produced by taking the hashed blocks, *media size*, *hash scheme*, and *hash size* as inputs to *H*(*x*).
media-hash
The length of a media-hash is also H_{len}.

Figure 1: The flow for generating hashed blocks and *media-hash*.

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 the sufficient information for a receiving peer to know (i) where the media object is stored; (ii) how to gather and stream the media object; (iii) what characteristics of a media object owns for decoding and rendering. Consequently, a *media-info* should be retrieved before a streaming session starts. Figure 2 shows the structure of a *media-info*. The fields of *media title*, *date time*, *creator*, and *comments* are extracted from original header of a media object, which are provided by *media-info* optionally. Note that *media-info* contains hashed blocks as well, which could be used to verify the correctness of each retrieved media block for the purpose of error recovery. Additionally, the field of brokers informs the receiving peer how to gather and stream the media blocks, which may probably spread among multiple peers. The detailed usage of this field is described in Section 3.3.

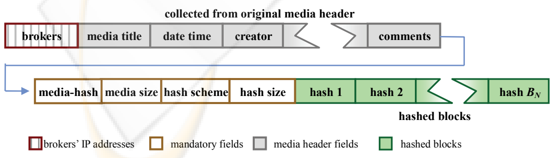

Figure 2: The structure of a *media-info*.

Especially, the size of the *media-info* highly depends on the choice of the size for each media block. Equation (1) expresses this relation, where M_{len}, B_{len}, H_{len} are the symbols as stated in Section 2.1, I_{len} is the size of *media-info*, and NF_{len}, OF_{len} represent the size of mandatory fields (including brokers field) and media header fields in Figure 2, respectively. Compact size of *media-info* tends toward larger B_{len}. However, larger B_{len} produces smaller amount of media blocks, which potentially reduces the degree of concurrence for media transmission among multiple peers.

$$I_{len} = \left\lceil \frac{M_{len}}{B_{len}} \right\rceil \times H_{len} + NF_{len} + OF_{len} \tag{1}$$

Based on the designed media representation, we now present the proposed architecture for providing VoD streaming services in a P2P environment.

3 MULTIOVERLAY ARCHITECTURE

In order to make VoD functions such as locate, stream, and control-playback for a media object behave smoothly with the abovementioned media representations, we devise the MegaDrop system, which has a multioverlay architecture with the four layers mentioned in Section 1, as shown in Figure 3.

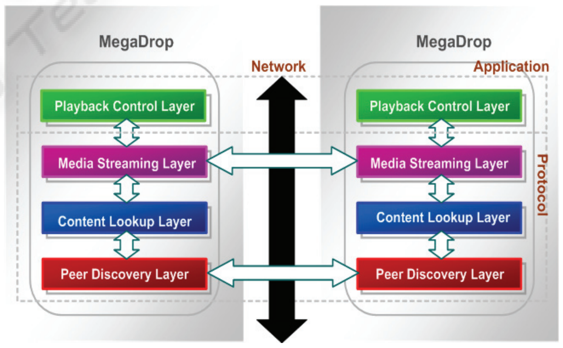

Figure 3: MegaDrop architecture.

3.1 Peer Discovery Layer

The PDL, as its name implies, provides an efficient peer lookup service such that upper layers can 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. Especially, The PDL is not restricted to a specific P2P network, that is, any typical P2P overlay networks such as Gnutella (Justine, 2000), Pastry

(Rowstron, 2001), Chord (Ion, 2001a), and CAN (Ion 2001b) can be adopted and wrapped in this layer as well.

3.2 Content Lookup Layer

The CLL plays the mediator for the PDL and the MSL. At least, two basic operations should be handled in this layer: (i) when a peer is willing to share a media object, a corresponding *media-info* has to be generated by the CLL, e.g., the CLL keeps a media pool for maintaining the *media-info*s of shared media objects; (ii) it should help content search for queries received by the PDL, so as to allow the PDL to focus on routing messages/queries to ignore processing the media content at all. Optionally, a mechanism for error recovery to ensure the completeness of received media blocks could be designed in this layer.

3.3 Media Streaming Layer

Media transmissions between peers are handled by this layer. Since the MSL has no idea of which peers media objects are located in, it must rely on the cooperation between the CLL and the PDL to find out these peers. In the MegaDrop system, overall peers could be classified as three types: suppliers, brokers, and droppers. A supplier is a peer that has a complete media object. A dropper, in contrast, is a peer that has an incomplete media object and still needs to retrieve missing media blocks from other peers. Therefore, for a specific media object, a supplier is always capable of providing any portions of media blocks, while a dropper only provide partial media blocks. Furthermore, a broker is an intermediate peer that assists suppliers and droppers in exchanging peer information (via the field of brokers in a *media-info* as mentioned in Section 2.2).

As Figure 4 displayed, during a media session in the MSL, several peers can form a cluster virtually, where some peers act as suppliers and the others act as droppers. Among these peers, one peer would be treated as a broker, which is connected by other peers to periodically maintain the list of sending peers. Note that in this virtual cluster, there should be existed at least either one supplier or several droppers containing exclusive media blocks to restore an entire media object.

Furthermore, providing buffer management for caching the most up-to-date media blocks of a media object is also necessary. In addition, concerning problems of fairness, or freeriding that exists in common P2P environments due to unequal contribution among peers, it is better to equip a bartering technique to raise incentive of contributing media objects by each peer.

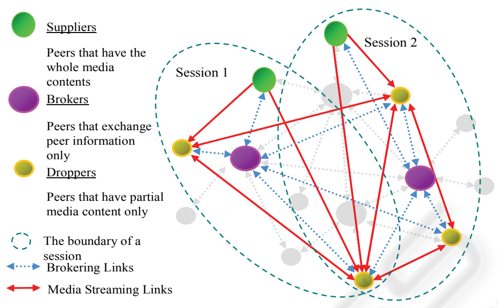
Suppliers
Peers that have the whole media contents
Brokers
Peers that exchange peer information only
Droppers
Peers that have partial media content only
The boundary of a session
Brokering Links
Media Streaming Links
Session 1
Session 2

Figure 4: The Media Streaming Layer.

3.4 Playback Control Layer

In VoD streaming services, the PCL is designed to directly interact with end users and exposes a user interface for administrating and controlling operations, e.g., providing VCR-like interactions, such as playing, pausing, stopping, seeking, fast-forwarding, and so on. Depending on users' behaviors, the PCL cooperates with the MSL to change caching policy (via buffer management). For example, when a user issue a seeking operation to change current play point, the PCL needs to notify the MSL to acquire the media blocks starting at the new play point first because the MSL never knows the occurrence of such an event without this notification. Besides, for the convenience of controlling playback, the PCL should apparently provide the status of monitoring and operational statistics for users to manage the overall system performance.

4 IMPLEMENTATION

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 from modifying an LGPL library, GnucDNA (GnucDNA, 2000), which is based on Gnutella2 (Gnutella2, 2005) overlay network. Figure 5 shows the topology. In experience, this topology benefits from minimizing searching and routing traffic to reserve network bandwidth for media transmission. Theoretically, the adequate amount of trunks could maintain the ability for a peer to find desired media blocks located anywhere on the network.

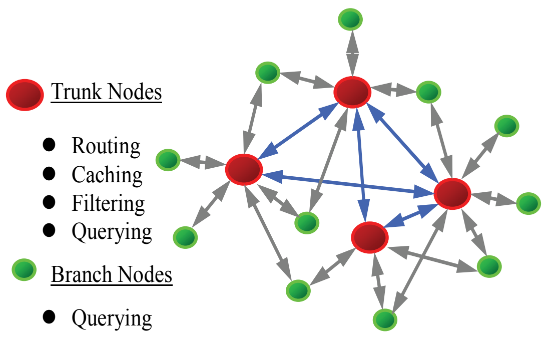

Figure 5: Overlay network with tree-like topology.

The following major operations are implemented in the PDL. First, bootstrapping allows a peer to participate in the MegaDrop system. We utilized GWebCache (GWebCache, 2003) to allow a peer to discover which active peers 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 protocol (Gnutella2, 2005). Likewise, query hash tables are utilized to facilitate the query-routing protocol, which is implemented by modifying Prinkey's scheme (Prinkey, 2001).

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. The first is generating *media-info*. We used the SHA1 algorithm as our hash scheme, which generated each hashed block in 20 bytes. The construction methodology is detailed in Section 2. 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 media according to the information provided by *media-info*. Thereafter, the CLL can cooperate 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 (Bram, 2001; Bram, 2003) to accomplish the most parts of this layer, and several operations are implemented: (i) *Media-info* requesting: the MSL needs this information before establishing a media session; (ii) Brokering: due to versatile status of active peers in P2P environment, a receiving peer can periodically update the list of sending peers from the broker through brokering mechanism. (Remind that a broker maintains a global view to aggregate a media object). This protocol is simply realized by HTTP conventions; (iii) Media streaming: this procedure enables a receiving peer maintaining TCP/IP connections to sending peers. In our implementation, this procedure contains three sub-functions, (a) handshaking, which is designed to ensure that whether a sending peer is serving the desired media object or not before a TCP/IP connection is established, (b) connection state guarding, which tracks remote peers' states (in three modes: chocking, unchoking, or interesting) to monitor this connection, and (c) message/media data communication, which delivers control or media data between both peers; (iv) Buffer management: in order to deliver media blocks efficiently, in the MSL, every media block can further decompose into a bunch of smaller units called media piece, such that a media piece could be treated as the smallest transmission unit during a media session. Therefore, a buffer space provided by sending and receiving peers could act as a cache to manage these media pieces. However, because media pieces of a media block may not always delivered in order or the play point of video may jump randomly due to users' behaviors, we used a request-on-demand policy to handle this issue; We omit the detail explanations here due to limit space; and last, (v) Bartering: a variant of tit-for-tat scheme is used to implement this technique. Finally, a typical access flow of the MSL is shown in Figure 6.

4.4 Playback Control Layer

We implemented the PCL via the Microsoft™ DirectShow media framework (Microsoft, 2006). DirectShow is essentially built on a group of filters, each of which performs a specific operation for streaming a media file. Connecting several filters via input/output ports results in 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. Figure 7 displays the topology, 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. This topology 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, we based them on filters provided by Microsoft™ SDK. Actually, we merely 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 was added to aid the MSL in gathering or splitting the media pieces. Finally, the overall VCR-like operations were implemented through the *IMediaControl* interface in DirectShow. Our implementation of PCL not only supports playback operations within the MegaDrop system, but also could be executed as a purely local media player.

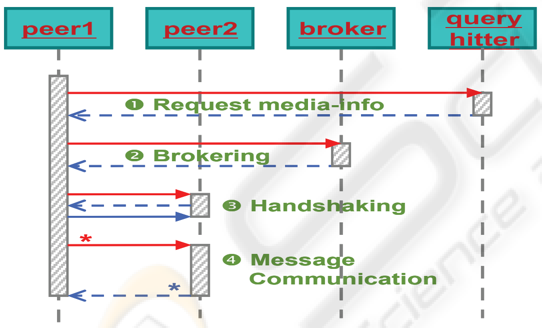

Figure 6: Access flows of the Media Streaming.

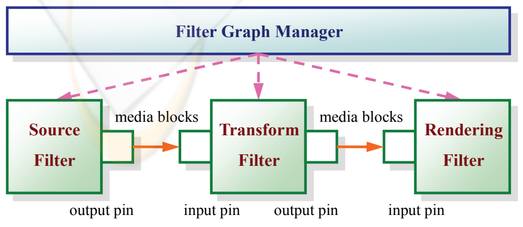

Figure 7: Topology of a DirectShow filter graph.

5 EVALUATION

In this section, we show that the MegaDrop system is particularly efficient of huge amount of media data and multiuser streaming sessions by evaluating it from three essential metrics: (i) the startup delay versus the size of the *media-info* before starting a media transmission; (ii) efficiency of bandwidth utilizations by measuring the overheads induced from network traffics other than real transmitted media blocks; and (iii) effectiveness of bandwidth aggregations among multiple peers.

5.1 Startup Delay

Delivering *media-info* merely relies on communication between a broker peer and the receiving peer, making it easy to measure the startup transmission delay. Assume that the maximum startup delay for a media session is D (in seconds) and the average network speed is BS (in bytes/second), from equation (1) listed in Section 2.2 we can derive the minimal size of the media block, B_{len}, required to satisfy the maximum tolerable startup delay D:

$$B_{len} \geq \frac{M_{len} \times H_{len}}{BS \times D - NF_{len} - OF_{len}} \qquad (2)$$

Figure 8 illustrates the relationship between the number of media blocks, the size of *media-info*, and media size for various media objects, and reveals that the length of media block should be carefully chosen based on the media size.

5.2 Efficiency of Bandwidth Utilization

The transmission of data in the system other than media blocks 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 Figure 9 indicate that the overhead ratio arises significantly only when the media object is smaller than 10 kB, but even then is still very low at less than 2.4%. Given that almost all media objects are significantly larger than 10 kB, the MegaDrop system exhibits high bandwidth utilization.

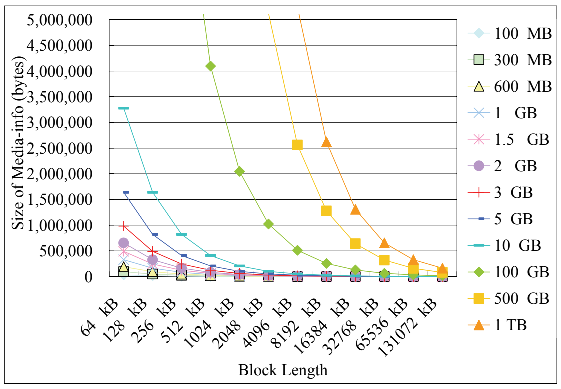

Figure 8: Size of *media-info* versus media block and media size.

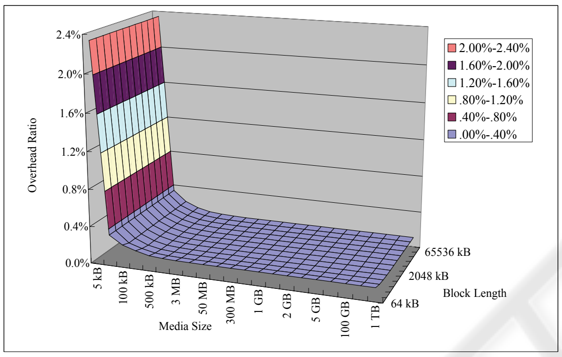

Figure 9: Overheads versus media size.

5.3 Effectiveness of Bandwidth Aggregation

To investigate effectiveness of bandwidth aggregating among peers, we setup an environment with four PCs in different specifications as shown in Table 1. Moreover, 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 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 will be triggered while bandwidth is running out. Therefore, during a media session, the join time of a peer, denoted as TJ_i, can be determined by equation (3), where N is the total number of peers in each test case.

$$TJ_i = \begin{cases} 0 & \text{, when } i = 1,2 \\ \dfrac{MLen \times (i-2)}{200 \times 1024 \times (N-1)} & \text{, when } 3 \le i \le N \end{cases} \quad (3)$$

Table 1: Specifications of the simulation environment.

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

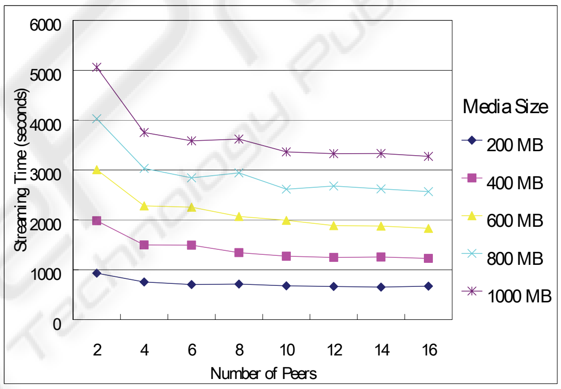

Figure 10: Average transmission time versus number of joined peers with various media size.

Figure 10 shows the results of the simulations. As we expected, the average transmission time per media session is reduced when more peers participated in it since this enlarged the degree of bandwidth aggregation. Especially for a large media object, the descending slope is more significantly but is retarded at a certain constant level, indicating that the effects of bandwidth aggregation are limited by the uplink bandwidth setting in the simulation scenario.

6 EXTENSIONS TO THE MEGADROP SYSTEM

Error recovery is implemented in the current MegaDrop system by simply dropping or retransmitting a received media block that is corrupted. As mentioned in Section 2, this simple scheme does prevent a video from being undecodable or damaged, but it may induce video glitches or longer transmission delay. To solve this problem, path diversity with *Multiple Description Coding* (MDC) technique (Frank, 2005; Ivan, 2005) can be adopted to combine with the originated media representations in the MegaDrop system. Basic idea is that, for a media object, every media block can be further decomposed into two or more sub media blocks based on MDC technique. For example, each media block can be divided into two sub media blocks, consisting even and odd video frames separately. Then in the MSL, every original media session can be split into multiple sub media sessions to deliver these sub media blocks accordingly through path diversity technique. Therefore, through this methodology, the probability of the occurrence in video glitches and longer transmission delay can greatly reduce by conceding to video quality. Besides, we leave several challenges such as security and QoS management for the readers, as the further researches to extend and consolidate the MegaDrop system.

7 CONCLUSIONS

In this paper we propose a novel layered architecture to realize VoD streaming services in a P2P environment. The proposed architecture implements a fully decentralized system running on a P2P multioverlay network. Unlike existing mechanisms in which certain video servers must be deployed in advance, the proposed architecture does not rely on any centralized resource allocation. Instead, every media object is delivered and propagated over the network, and every peer in the network retrieves media content from as well as forwards it to other peers, thereby acting as a miniserver. The processing and sharing of the same media objects by multiple peers results in the formation of virtual server clusters. Major advantages of the proposed architecture are that it can balance the load among peers and efficiently utilize the network bandwidth.

The experimental results revealed that our approach is appropriate for serving VoD streaming, especially in the delivery of huge amounts of media data. Moreover, the participation of more peers in a media session can result in higher bandwidth aggregation, result in a decrease in the average time required to stream a given media object.

Finally, we have presented methods for reducing the probabilities of video glitches and longer transmission delays by combining our original media representations with the promising MDC and path-diversity techniques. These concepts are recommended as topics for future research to extend and consolidate the MegaDrop system.

REFERENCES

Bram Cohen (2001), *BitTorrent Protocol*, [Online], Available: http://www.bittorrent.com/protocol.html.

Bram Cohen (2003) 'Incentives build robustness in BitTorrent', in *Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems*, Berkeley, CA.

Christofer Rohrs (2001) 'Query routing for the Gnutella network', [Online], Available: http://rfc-gnutella.sourceforge.net/.

GnucDNA (2000), [Online], Available: http://www.gnucleus.com/GnucDNA/.

Gnutella2 (2005), [Online], Available: http://www.gnutella2.com/.

GWebCache (2003) 'Gnutella Web Caching System, [Online], Available: http://www.gnucleus.com/gwebcache/.

Frank. H.P. Fitzek, Basak Can, Ramjee Prasad and Marcos Katz (2005) 'Traffic analysis and video quality evaluation of multiple description coded video services for fourth generation wireless IP networks', in *Special Issue of the International Journal on Wireless Personal Communications*.

Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek and Hari Balakrishnan (2001a) 'Chord: A scalable Peer-to-peer Lookup Service for Internet Applications', in *Proceedings of ACM SIGCOMM 2001*, San Diego, CA, USA.

Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan (2001b) 'Chord: A scalable contentaddressable network', in Proceedings of the ACM SIGCOMM 2001, San Diego, CA, USA.

Ivan Lee and Ling Guan (2005) 'Reliable Video Communication with Multi-Path Streaming Using MDC', in *Proceedings of the IEEE International Conference on Multimedia & Expo*, pp. 711-714.

Justine Frankel and Tom Pepper (2000) *Gnutella*, [Online], Available: http://www.gnutella.com/.

Kien A. Hua, and Mounir Tantaoui (2003) 'Cost effective and scalable video streaming techniques', in B. Furht and O. Marques (ed.) *Handbook of Video Databases*, Abingdon: CRC Press.

Microsoft Corporation (2006) 'Microsoft Developer Network (MSDN)', [Online], Available: http://msdn.microsoft.com/.

Simon Sheu, Kien A. Hua and Wallapak Tavanapong, (1997) 'Chaining: A generalized batching technique for Video-on-Demand Systems', in *Proceedings of the IEEE International Conference on Multimedia Computing and System,* Ottawa, Canada, pp. 110-117.

Yang Guo, Kyoungwon Suh, Jim Kurose and Don Towsley (2003a) 'A Peer-to-peer On-Demand streaming service and its performance evaluation', in *Proceedings of the IEEE International Conference on Multimedia & Expo*, Baltimore, MD, pp. II-649-652.

Yang Guo, Kyoungwon Suh, Jim Kurose and Don Towsley (2003b) 'P2Cast: Peer-to-Peer patching scheme for VoD service', in *Proceeding of the 12th International World Wide Web Conference*, Budapest, Hungary, pp. 301-309.

Tai T. Do, Kien A. Hua and Mounir Tantaoui (2004) 'P2VoD: Providing fault tolerant Video-on-Demand streaming in Peer-to-peer environment', in *Proceedings of IEEE International Conference on Communications*, Paris, pp.1467-1472.

Rowstron A. and Druschel P. (2001) 'Pastry: scalable, decentralized object location and routing for large-scale Peer-to-peer systems', in *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms,* Heidelberg, Germany.

Prinkey M. T. (2001) 'An efficient scheme for query processing on peer-to-peer networks', [Online], Available: http://aeolusres.homestead.com/files/index.html