Distributed Storage-Assisted Data-Driven Overlay Network for P2P VoD Services

Changqiao Xu, Gabriel-Miro Muntean, *Member, IEEE*, Enda Fallon, and Austin Hanley

***Abstract*—Providing VCR-like operations in Peer-to-Peer (P2P) environments is a significant challenge. This paper proposes a distributed Storage-assisted Data-driven overlay Network (SDNet) to support P2P Video-on-Demand (VoD) services. It integrates two networks: a Data-driven Overlay Network (DONet) and a multi-way tree. DONet is enhanced and used for the routine video distribution based on the buffer overlapping mechanism and gossip protocol. A novel algorithm which uses a multi-way tree structure and extra pre-fetching buffers at the nodes is proposed to support efficient VoD operations. Videos are divided into uniform segments, pre-fetched and stored in a distributed manner along the tree topology. The cooperation between DONet-based video delivery and the tree-located multimedia components enable multimedia streaming interactive commands to be performed efficiently. This paper presents and discusses the structure of SDNet and the distributed storage scheme and details the cooperation procedure. Simulation-based testing results show how the proposed SDNet is an efficient interactive streaming solution in a P2P environment.**

***Index Terms*—DONet, multi-way tree, P2P VoD, VCR-like.**

I. Introduction

PEER-TO-PEER (P2P) multimedia streaming has been studied extensively recently [1]–[18]. The research work on P2P streaming can be classified into P2P live streaming [1]–[4] and P2P Video-on-Demand (VoD) [5]–[14]. In most of these research papers, the authors always assume that the peers start playback from the beginning or from the current point of streaming when they join the streaming session, and the peers will keep on watching until they leave or fail the session. This assumption ignores the users' interactivity [19], [20]. Based on a large number of observations for true VoD operations, [21], [22] it was noted that: 1) most multimedia objects are visited partially; 2) many VCR-like operations such as forward, backward, random-seek occur, as they are very popular when people watch multimedia programs. Providing this level of interactive streaming service in a P2P environment is a significant challenge. In P2P live streaming, all of the peers have a "close" playback time: a peer can find its streaming suppliers easily when it joins the streaming session. However, P2P VoD streaming allows the peers to request streaming asynchronously with different offsets, so it is difficult to find suitable streaming suppliers. Although media files could be downloaded in advance using P2P file transfer technologies such as BitTorrent [23] and played it on-demand afterwards, this approach introduces long startup delay for playback which are not acceptable.

The authors of [24], [25] proposed patching and batching schemes respectively to support near VCR-like functions for VoD delivery. However, these are solutions for a traditional client-server framework and both the network and I/O bandwidth of the server can become bottlenecks easily, so the scalability of the resulting VoD system is limited.

A "cache-and-relay" mechanism [15] is used in P2P streaming to cache the played content and relay it to the peers who request the same streaming. In this context, the P2P on-demand overlay streaming system oStream [5] employs a pure tree structure, in which each node caches played out data and relays to its children at asynchronous playback times. A centralized directory server is used to maintain the global information of the overlay, which facilitates node join or failure recovery. Furthermore, the "cache-and-relay" mechanism is extended in [12], [16], [17] by caching only the played portion and pre-fetch some stream segments in advance using additional bandwidth (and storage). The pre-fetching strategy can grant peers the ability to overcome the bursty packet loss, the departure of a source peer, and to enhance the playing experience.

Most research use pre-fetching strategy to assist solving the interactivity problem for multimedia streaming. Researchers have proposed a hierarchical pre-fetching scheme for popular or sub-popular segments in [6], which is based on the assumption that the popular or sub-popular segments will be visited more often with random-seek. However, this scheme's main challenge is collecting the user viewing logs in a distributed P2P system, which is different from the traditional client-server network in which there is a central server which can support this. A similar solution is proposed in [26].

In VMesh [7], [8], video segments are pre-fetched and stored in nodes over a Chord [27] network. Each node maintains previous/next-segment-lists based on distributed hash tables (DHT) and uses these lists to support short jumps. If the jump is too far away, a DHT search is triggered for the segment corresponding to the new position. VMesh does not use a "cache-and-relay" mechanism, so the video segments in playback buffer are discarded and not made full use of. DHT is efficient for exact queries but is not well suited for range queries since hashing destroys the ordering of data.

Manuscript received February 29, 2008; revised July 16, 2008. First version published December 02, 2008; current version published February 25, 2009.

C. Q. Xu is with the Software Research Centre, Athlone Institute of Technology, Athlone, Ireland. He is also with the Institute of Software, Chinese Academy of Sciences, Beijing, China, and with the Graduate University of the Chinese Academy of Sciences, Beijing, China (e-mail: cqiaoxu@gmail.com).

G.-M. Muntean is with the Performance Engineering Laboratory, School of Electronic Engineering, Dublin City University, Glasnevin, Dublin 9, Ireland (e-mail: munteang@eeng.dcu.ie).

E. Fallon and A. Hanley are with the Software Research Centre, Athlone Institute of Technology, Athlone, Ireland (e-mail: efallon@ait.ie; ahanley@ait.ie).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBC.2008.2006252

In RINDY [9], each peer maintains a set of concentric rings and places all neighbors on these rings according to their present playing distances. It uses gossip messages between neighbors of the rings to locate the nodes which have the content sought. Its main problems are the increase of both jump latency and failure rate. If a node A intending to jump forward within the video stream locates a node B which has the segment corresponding to the new position and at that time B initiates a jump as well, then A';s jump fails. Consequently search for another node is required, which increases significantly the jump latency. If all the nodes have "close" playback time, then the number of rings each node maintains is small, which eventually determines an increase in the jump failure rate.

The authors of [10] proposed a pre-fetching algorithm based on a balanced binary tree which was deployed successfully to an unstructured media dissemination network [11]. During the resulting Balanced Binary Tree-based Unstructured VoD distribution (BBTU) [11], the pre-fetching buffer will supply the multimedia streaming temporarily for any node's jump, which decreases the latency for the jump, reduces the impact of the target node jumps and ensures a high jump success rate.

Mesh-based overlay multicast is considered as a better choice to support live streaming. Data-driven Overlay Network (DONet) is a representative solution for this kind of protocols [1]. In DONet, the stream propagated in the overlay network is divided into multiple segments. Each node maintains a number of segments, and exchanges the buffer map of available segments with partner nodes. After learning the buffer maps of partner nodes, each node requests a certain segment from a suitable partner node that holds the segment. If a node receives segment requests from partner nodes, it replies to the requests by forwarding the corresponding segments within its outgoing bandwidth. Therefore, the leave or crash of a single node will not impact too much on other nodes. DONet uses successfully a random gossip algorithm [28] for scalable live video streaming, but it does not support VCR-like operations commonly encountered in VoD service distribution.

This paper proposes a distributed Storage-assisted Data-driven overlay Network (SDNet) to address VoD VCR-like operations-related issues and to increase their efficiency. SDNet uses a multi-way tree structure of additional pre-fetching buffers at its nodes. Video segments are pre-fetched and stored in the nodes' pre-fetching buffers along the tree in a distributed manner, which will support interactivity. Furthermore, SDNet enhances DONet to distribute streaming content between nodes by using the buffer overlapping mechanism and gossip protocol. The paper presents and discusses SDNet's associated algorithms and evaluates the performance of the proposed solution.

II. SDNet Architecture and Principle

SDNet has a hybrid architecture illustrated in Fig. 1, which integrates two networks: a multi-way tree and DONet. The multi-way tree is based on BATON*, which is presented in [18], paper which also shows how this balanced tree is constructed. In BATON*, each node has three different kinds of links: parent links pointing to parent nodes, children links pointing to child nodes and adjacent links pointing to adjacent (in linear order) nodes. The most significant aspect is that the cost of a search operation in BATON* is bounded by $O(\log_m N)$.

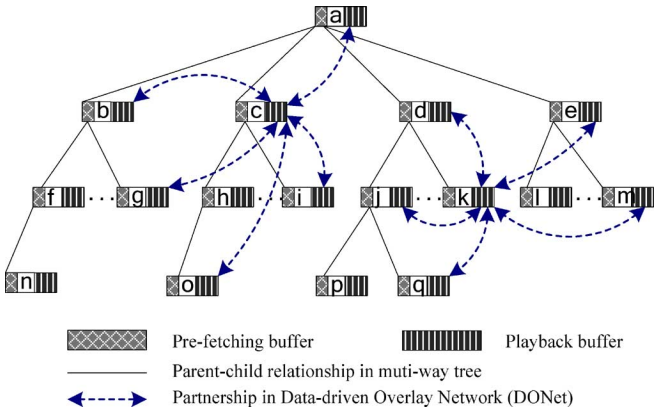


Fig. 1. The architecture of SDNet.

Apart from the playback buffer used during video streaming, SDNet involves the addition of an extra pre-fetching buffer to each node. Videos are divided into uniform segments. Upon entering the system, a new client may start viewing on demand a video stream. Using its residual bandwidth, it also downloads in advance some video segments and stores them in its pre-fetching buffer. The video segments are not selected at random; they are chosen by the pre-fetching scheme, which orders the video segments in the adjacent nodes' pre-fetching buffers according to the playback time. After the video segments are completely downloaded, they will be stored during the node's lifetime.

Although existing DONet-based approach is successful for scalable live video streaming, this pure mesh overlay network-based solution may lead to unacceptable latency or even failure of VCR-like operations for VoD services where nodes usually have different playing offsets, across a wide range. In SDNet, DONet was enhanced to support VoD delivery by deploying a buffer overlapping mechanism and by using a gossip protocol. The enhanced DONet is used as the routine overlay for video dissemination.

Under normal conditions, each node gets the multimedia streaming from its gossip partners in DONet. When VCR-like operations occur, the partners updating progress is triggered in DONet and during its course, the multi-way tree assists the node to jump to the new playing scene quickly. The adjacent nodes in the tree will provide the node with the multimedia streaming segments from their pre-fetching buffers. After re-establishing new partners in DONet, the streaming suppliers are switched to the new partners again.

SDNet has the following main features: firstly, it integrates balanced multi-way tree's high search efficiency which decreases the latency for VCR-like operations; secondly, it inherits good robustness from the random gossip protocol in DONet and the merit of "cache-and-relay" mechanism; thirdly, the stable video segments in nodes' pre-fetching buffer achieve the high success rate for random-seek and enhance the multimedia streaming playing experience.

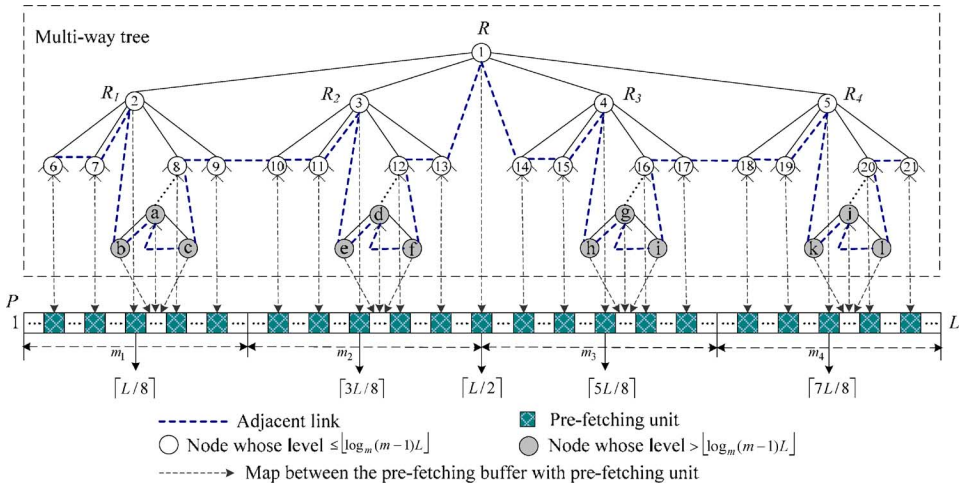


Fig. 2. Pre-fetching scheme.

TABLE I
NOTATION USED OF PRE-FETCHING ALGORITHM

Notation	Descriptions
S	The VoD media server
T	The multi-way tree
P	The requested video for playing
Len	Length of *P* (the total seconds time for playback)
d	Size of one pre-fetching unit
R	The root node of *T*
X	A node in *T*
Level(*X*)	The level of *X* in *T*
Prebuf(*X*)	Node *X*′s pre-fetching buffer
Preunit(*X*)	The sequence number of pre-fetching unit in *Prebuf*(*X*)
Parent(*X*)	Node *X*′s parent node in *T*

III. SDNET PRE-FETCHING SCHEME

Assuming that the notation presented in Table I are used, if the video P is coded using CBR rate and is divided into uniform segments (e.g. one segment is 1 sec. playback length), d numbers of successive segments are set as one pre-fetching unit, then P has $L = \lceil Len/d \rceil$ pre-fetching units which are numbered from 1 to L sequentially. The pre-fetching buffer of node X is defined as $Prebuf(X)$ with size of d video segments. Fanout of the tree T is set as even, a range of values managed by a node is greater than ranges of values managed by the first $\lceil m/2 \rceil$ children nodes while less than ranges of values managed by the last $\lfloor m/2 \rfloor$ children nodes. The proposed pre-fetching scheme is illustrated in Fig. 2. As it shows, at the level 0 of T, the mapping relationship between the corresponding pre-fetching unit of P's middle position is set with $Prebuf(R)$, namely $Preunit(R) = \lceil L/2 \rceil$. At level 1, the children nodes (from left to right) of R are named $R_1, R_2, \ldots, R_m$ and P is divided into m equal subsections $L_1, L_2, \ldots, L_m$. Assuming $Mid(L_1), Mid(L_2), \ldots, Mid(L_m)$ are the pre-fetching units' sequence numbers corresponding to the middle position of $L_1, L_2, \ldots, L_m$ respectively, $Preunit(R_1) = Mid(L_1), Preunit(R_2) = Mid(L_2), \ldots, Preunit(R_m) = Mid(L_m)$ are set. The above operations are repeated for each of the tree levels until P can not be divided anymore (the subsection length is less than one pre-fetching unit). Then, the pre-fetching unit's sequence number is set to equal its parent node's.

In Fig. 2, '1', '2', ..., 'k', 'l' are the node names (not pre-fetching unit sequence numbers). The line linking node '8' and 'a' is dashed suggesting that many nodes are actually omitted between the level corresponding to node '8' and the level corresponding to node 'a'. As nodes 'a', 'b', 'c' have the same pre-fetching unit, the arrow originating from nodes 'b' and 'c' points to the same direction as the arrow of node 'a'. For node X, presuming X is the ith child of $Parent(X)$, $Level(R) = 0$. From the settings and notation presented above, the following equations can be deduced:

1) If $Level(X) = 0$, then

$$Preunit(X) = \lceil Len/2d \rceil \tag{1}$$

2) If $0 < Level(X) \leq \lfloor \log_m(m-1)Len/d \rfloor$, there are two cases:
   - If $1 \leq i \leq \lceil m/2 \rceil$, then

$$Preunit(X) = Preunit\,(Parent(X)) - \left\lceil \frac{(m-2i+1)Len}{2m^{Level(X)}d} \right\rceil \tag{2}$$

   - If $\lceil m/2 \rceil < i \leq m$, then

$$Preunit(X) = Preunit\,(Parent(X)) + \left\lceil \frac{(2i-m-1)Len}{2m^{Level(X)}d} \right\rceil \tag{3}$$

3) If $Level(X) > \lfloor \log_m(m-1)Len/d \rfloor$, then

$$Preunit(X) = Preunit\,(Parent(X)) \tag{4}$$

For conditions (1)–(3), X needs pre-fetching of the specific pre-fetching unit from S; for condition (4), X gets the pre-fetching unit from its $Parent(X)$ or sibling nodes or one of its adjacent nodes whose pre-fetching unit sequence number equals $Preunit(X)$. Due to (4) downloading all the pre-fetching units from S is avoided, alleviating the load of S. A list of peers which store the same pre-fetching unit is maintained in order to support load balancing.

BATON* is adaptive to dynamic network, for node join, departure, failure. If a node's position in T is changed, it should

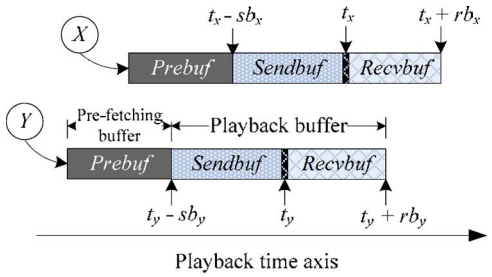


Fig. 3. Node buffer status at time t.

check whether the pre-fetching unit in its pre-fetching buffer satisfies the (1)–(4); if not it should update the pre-fetching unit.

IV. Enhanced DONet for VoD

A. Buffer Overlapping Mechanism

In P2P environment, for any node X and Y, assuming they request the same streaming, the video segments aggregate in X and Y's playback buffer are defined as $C(X)$, $C(Y)$ respectively. If $C(X) \cap C(Y) \neq \phi$, it means they are "close" enough in playback time, their playback buffer overlaps, so node X can supply the streaming service to Y and Y does not need to get the streaming from S. This lightens the load of S. Note that as long as no VCR-like operations occur and the network bandwidth is sufficient, this buffer overlapping relationship is unchanged. The buffer is divided into three parts at every node X: the pre-fetching buffer which has been discussed above, buffer named $Recvbuf(X)$ with size rb_x, which stores data segments pre-fetched from other nodes, and sending buffer named $Sendbuf(X)$ with size sb_x, which caches played out segments, both of which can be used to supply other nodes upon request. The receiving buffer and sending buffer form the playback buffer.

Fig. 3 depicts a snapshot of the buffers at nodes X and Y respectively at time t. Supposing t_x, t_y is the currently played segment for node X, Y respectively, the minimum serial number of the video segment in $Sendbuf(X)$ is $t_x - sb_x$ and the maximum serial number in $Recvbuf(X)$ is $t_x + rb_x$. A similar situation can be derived in relation to node Y.

Assuming $C(Sb_x)$, $C(Rb_x)$, $C(Rb_y)$ are the video segments set in $Sendbuf(X)$, $Recvbuf(X)$, $Recvbuf(Y)$, if $(C(Sb_x) \cup C(Rb_x)) \cap C(Rb_y) \neq \phi$ then X can be the streaming supplier for node Y. Consequently (5) is satisfied:

$$\begin{cases} t_x - sb_x < t_y + rb_y \\ t_x + rb_x > t_y \end{cases} \tag{5}$$

Similarly, Y can be a streaming supplier of X if it satisfies (6):

$$\begin{cases} t_y - sb_y < t_x + rb_x \\ t_y + rb_y > t_x \end{cases} \tag{6}$$

By combining (5) and (6), (7) can be deduced:

$$t_x - rb_y < t_y < t_x + rb_x \tag{7}$$

This result describes a node's playback window status information and gives the premise conditions for node X and Y to establish partnership, which will be employed to search partners in DONet.

B. Enhanced DONet

In DONet, each node maintains a membership cache (*mCache*) containing a partial list of the identifiers for the active nodes in the DONet. The *mCache* assists to establish the partnership with other nodes, each node selecting some random nodes in its mCache and trying to establish partnership with the nodes. Consequently the key practical issue is how to maintain and update the *mCache*. It is known that each node in DONet periodically generates a membership gossip message which is sent over the other nodes to announce it existence. The message includes:

$$\langle seq\_num, id, num\_partner, time\_to\_live \rangle$$

which is sufficient and efficient for live video streaming.

However, for VoD delivery, nodes usually have different playing offsets across a wide range and only the nodes whose playback buffers overlap can be partners. Consequently it is imperative for the node's buffer window status to be integrated into the message, so in SDNet, the gossip message is extended to

$$\langle seq\_num, id, num\_partner, time\_to\_live, current, min, max \rangle$$

where seq_num is a sequence number of the message, id is the node's unique identifier, $num\_partner$ is its current number of partners, and $time\_to\_live$ records the remaining hop count of this message. $current$, min and max denote a snapshot of the moving playback buffer window; they represent the current playing position, minimal segment and maximal segment of the playback window respectively. In SDNet, each peer periodically sends the gossip message with an initial $time\_to\_live$ to some random nodes in its *mCache*. Then these nodes select one random node from their *mCache*, decrease its $time\_to\_live$ and forward this message to the selected node. This procedure is iterated until the $time\_to\_live$ becomes zero or the selected node is the source of this message. When the node receives the gossip messages, it takes the information from the messages and tests the condition indicated in (7). If the condition is satisfied, the node updates its *mCache*.

When VCR-like operations occur in a node, partners updating process needs to be supported by SDNet. This is because the node's playing position changes and the old partners cannot provide the node's new streaming content. Two message types *REQUEST* and *REPLY* are proposed to be used in order to locate new partners required by jump operations during the partners updating process. The message format is

$$\langle typeid, segid, srcid, time\_to\_live, routepath, resultset \rangle$$

where $typeid$ represents the type of message (*REQUEST* or *REPLY*); $segid$ is the target segment sequence number corre-

sponding to the new jump position; $srcid$ denotes the source address of the message; $time\_to\_live$ is the remaining hop count of this message; $routepath$ is a list of peers traversed by the request message and $resultset$ stores the result of the query. Assuming that X triggers partners updating process, X pushes its address into the $routepath$, then forwards this message to one random node in its *mCache*. Upon receipt of this query, the visited node executes the same procedure, until this request arrives at the node whose playback buffer includes the segment whose sequence number equals $segid$, then the peer adds all of its partners into the $resultset$, changes this message to *REPLY*, and returns the message to X along the routing path. When X receives the *REPLY* message, it adds the nodes in $resultset$ to a candidate nodes list and sends a gossip message including X's present playback window status information to each candidate node. When the candidate node receives the message, it tests condition (7) and if it is satisfied, the candidate node returns the HconfirmationH message and X adds the candidate node into its gossip partners list. If X cannot find the target node until $time\_to\_live = 0$, X sends the *REQUEST* message to another node in its *mCache* and tries again to find its new gossip partners.

V. Supporting VCR-Like Operations

When VCR-like operations occur for node X during its playback (e.g. random-seek), the target video segment corresponding to the new position can be calculated roughly by the player interface. Assuming it is $segid$, the corresponding pre-fetching unit is $\lceil segid/d \rceil$, which is denoted with M. In SDNet, each node maintains a queue called *Que* with size of K (invariable parameter), *Que* is like a dynamic routing table and each item from *Que* keeps a reference (i.e., the IP address) pointing to a node. The VCR operations supporting procedure is as follows:

1) X empties its partners list, partners updating progress is triggered in the enhanced DONet using the mechanism discussed in Section IV-B;
2) The pre-fetching unit of X's left or right adjacent node is compared with M, if it is equal, then the left or right adjacent node of X is regarded as node J and the algorithm continue with step 5). Otherwise the next step is executed;
3) T is traversed begin with R, M is compared with $Preunit(R)$, if equals, then R is the target node, R is regarded as node J, jumps to step 5), otherwise, executes next step;
4) T is traversed until node J with $Preunit(J) = M$. If J can not be found until node H with $Level(H) > \lfloor \log_m(m-1)Len/d \rfloor$ is visited, failure is returned;
5) Assuming Ln is the tail node of *Que*, Cn is the current visited node, J is put into *Que* and the right adjacent link of J begins to be traversed. If $Preunit(Cn) = Preunit(Ln) + 1$, Cn is put into *Que*. If $Preunit(Cn) = Preunit(Ln)$ and the present usable bandwidth of Cn is more than Ln's, then Ln is replaced by Cn. If K' (invariable parameter, default value is 5) numbers of successive visited nodes have the same pre-fetching unit, then Cn switches to R directly and searches from R to locate the node Y with $Preunit(Y) = Preunit(Ln) + 1$, then continues to traverse node Y's right adjacent node. The traversing progress repeats until finding the node whose pre-fetching unit sequence number equals $Preunit(Ln) + 1$ fails or *Que* is full;
6) The head of *Que* begins to send its video segments in its pre-fetching buffer beginning with the video segment whose sequence number is $segid$. The other nodes in *Que* send all segments in their pre-fetching buffer to X in order. This streaming transmission sequence is just the continuous video segments begin with $segid$ for the new playing position. The node which finishes sending its pre-fetching video segments is deleted from *Que*;
7) When X finishes re-establishing its new partners, *Que* is notified to stop sending streaming. Then, the multimedia streaming suppliers are switched to X's new partners. Otherwise, if there are only K" (invariable parameter) numbers of nodes in *Que*, the right adjacent node of Ln is regarded as J and executes the step 5) again, then new nodes are put into *Que* and *Que* continues to send streaming to X, this operation repeats until X finishes re-establishing. At worst, if X can not re-establish its new partners successfully, *Que* will supply X with the continuous multimedia streaming, which ensures the success and stability for random-seek.

As Fig. 4 shows, presuming node 3 jumps during playback, the video segment corresponding to the new target playback position is g which can be calculated approximately by the player interface. After searching in T, it is presumed that g falls into node 4's pre-fetching buffer. Node 3 empties its partner's list which includes nodes 1, 2, 9 and k and triggers partners updating process in SDNet. During this process, node 4 and nodes o, 16, 17, 18 (which are the traversal results after executing step 4) send its pre-fetching buffer's video segments to node 3 until node 3 has established its new partnership nodes (which are 6, g, j, 13). Then the new partner nodes will supply the streaming to node 3.

VI. Performance Evaluation

A. Simulation Settings

The source media server has ten videos for streaming, each with 256 Kbps rate and two hours length. A segment length for playback is 1 sec., the playback buffer can accommodate 720 segments and is divided equally into receiving buffer and sending buffer. The default size of a pre-fetching unit is 30 segments. The default number of gossip partners for each node is 6. The performance of SDNet is evaluated with various parameter settings, and also is compared with VMesh. VMesh is built on top of a public Chord implementation [29]; the length and bit rate of the video and each pre-fetching unit's length are set same as SDNet's. The topology is generated using the GT-ITM package [30], which emulates the hierarchical structure of the internet by composing interconnected transit and stub domains. The network topology for the presented results consists of ten transit domains, each with twelve transit nodes, and a transit node is then connected to six stub domains, each with nine stub nodes. The total number of nodes is thus 6,600. It is assumed

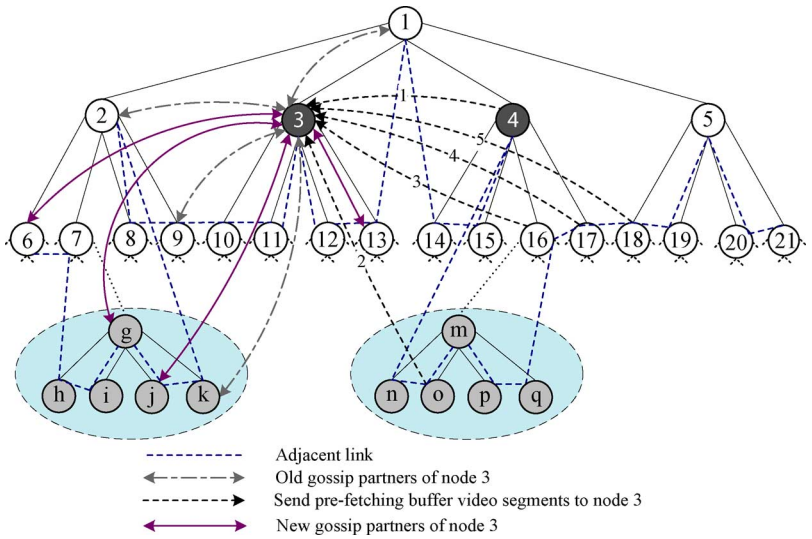


Fig. 4. VCR-like operations supporting procedure.

that each node represents a local area network with plenty of bandwidth, and routing between two nodes in the network follows the shortest path.

The initial bandwidth assigned to the links is as follows: 1.5 Mbps between two stub nodes, 6 Mbps between a stub node and a transit node, and 10 Mbps between two transit nodes. CBR over UDP cross traffic is generated and transmitted over the network links such that the available bandwidth at each link varies over time. This is used during the experiments in order to emulate dynamic network conditions. The peers participating in the system follow a Poisson arrival. To mitigate randomness, each result presented in this section is the average over ten runs of the same experiment.

B. Performance Comparison

1) Jump Latency: Jump latency is the time period starting from the moment when a client issues a jump command to the instant that the client starts playing out at the new position. In SDNet when a jump occurs, the pre-fetching buffer will be the source of streaming temporarily before the node finishes re-establishing its new partner list. In this context the jump latency is assessed in terms of the hop count for locating the node in the tree [7]. The segment corresponding to the new position falls into this node's pre-fetching buffer. The node's right adjacent nodes will constitute the successive streaming for the new scene using their pre-fetching buffers.

The jump latency of SDNet is evaluated in comparison with VMesh. Fig. 5 presents a comparison between results when different parameters are used: the size of pre-fetching unit d and the number of nodes in system are set to different values. These results show how SDNet is more efficient than VMesh. In VMesh, the cost for search via DHT is $O(\log N)$, where N is the number of nodes in system and the cost increases with the increasing of total number of nodes. Although the cost for search in BATON* is $O(\log_m N)$, taking SDNet's pre-fetching scheme into account, the hops for search a target node which includes any playing position segments are less than $O(\log_m(m-1)\lceil Len/d\rceil)$. This cost decreases with the increasing of pre-fetching unit size.

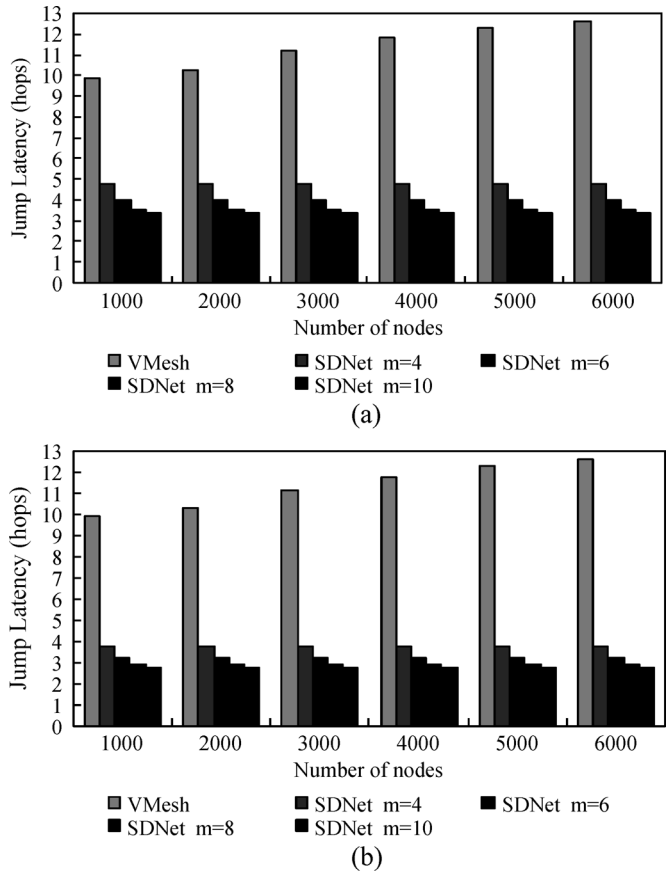


Fig. 5. Jump latency: (a) $d = 30$; (b) $d = 120$.

2) Jump Failure: It is considered that a jump fails when a node cannot locate a target node which has the segment corresponding to the new position during its jump. Jump failure rate (JFR) is computed as the average ratio between the total numbers of nodes which fail to jump and the total number of nodes which have requested to jump during playback. JFR of SDNet

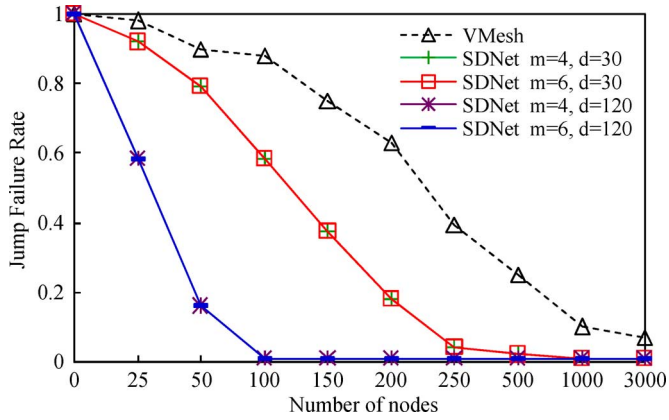


Fig. 6. Jump failure.

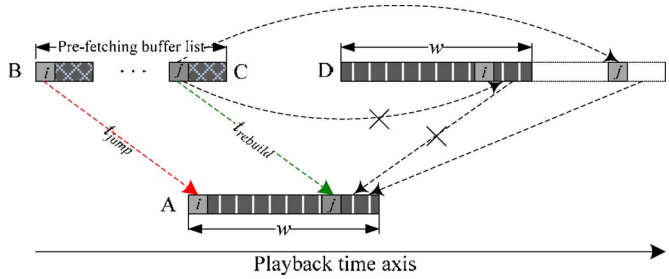


Fig. 7. The impact of playback window size.

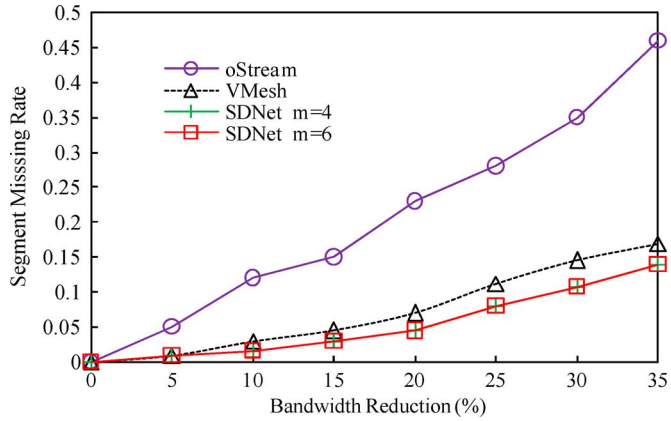


Fig. 8. The impact of dynamic network.

is compared with that of VMesh when random-seek occurs. JFR reflects the capability to support the random-seek function. Fig. 6 shows how the JFR of the two systems decreases with the increase in the number of nodes. However the decreasing rate for VMesh is generally lower than that of SDNet. This is because in VMesh, the distributed hash table (DHT) is utilized for segments' storage in nodes, which leads to the independence of segment storage for each node (the hashing destroys the ordering of data). In SDNet, a peer fetches those segments suggested by the pre-fetching algorithm, which enables the distribution of the whole segments regularly over nodes. In general case, $\lceil Len/d \rceil$ numbers of nodes' pre-fetching buffers can share the whole video segments of the requested video clip.

In SDNet, though the pre-fetching buffer ensures the high success rate for jump, the video segments in pre-fetching buffer are provisional suppliers and it is not good that one node gets the streaming from other nodes' pre-fetching buffer for a long time, because they also need to assist other nodes to jump. So the node needs to establish new partners in DONet as soon as possible. As Fig. 7 shows, assuming w is the playback window size, node A initiates jump at time t_{jump}, the segment of the new position is i. During A updating its gossip partners, nodes B and C provide streaming for A using their pre-fetching buffers. At time $t_{rebuild}$, D is found and its playback buffer includes i. Assuming j is the current playing segment in A, the interval between the jump time to the time that D has been found is $\Delta t = t_{rebuild} - t_{jump}$, so the sequence number of segment j should be $i + \Delta t$. Although D includes segment i, it is possible that segment i is just put into D's playback buffer and j has not arrived and falls out of D's playback window. If condition (7) is tested between A and D with each nodes' current playback window status values, it may not be satisfied, so D becoming As gossip partner fails actually. Consequently, the possibility that D's existing partners become A's partners is low. If w is strengthened, it is likely that D becomes A's partner, so w should be set to an appropriate value to increase the success rate for establishing gossip partners in DONet.

3) Streaming Quality: To playback continuously and smoothly is important for P2P streaming applications. The Segment Missing Rate (SMR) is adopted as the metric to evaluate video streaming quality. A data segment is considered missing if it is not available at a node before the play-out time. SMR for the whole system is the average of the SMRs computed at all the participating nodes during the simulation time. As such, SMR reflects two important aspects of the system performance: delay and capacity. For comparison, the existing on-demand overlay streaming system, oStream is simulated with the same network and buffer settings.

Firstly, the performance under dynamic network environments with bandwidth fluctuations is investigated. To emulate bandwidth fluctuations, the bandwidth is decreased from 100% to 65% of the base setting. As Fig. 8 shows and as expected, the SMRs of SDNet, VMesh and oStream increase with the decrease in the bandwidth. However, the increasing rates for VMesh and SDNet are much lower than that of oStream. In particular SDNet's SMR increasing rate is lower than that of VMesh.

Secondly, the performance under random-seek with seeking rate fluctuations is studied. For oStream, random-seek can be implemented by letting the node leave the system and then rejoin with the new playback offset. The random-seek rate is defined as the average ratio of the total nodes which random seek to the total nodes of the whole system. As Fig. 9 shows, when 8% nodes of system random seek, the SMR of SDNet is less than 10%, the SMR of VMesh is close to 10% and the SMR of oStream has reached 30%. For SDNet, SMR decreases with the increasing fanout of the tree, because the bigger fanout achieves the more efficiency for random-seek.

In oStream, once a segment is lost at a high level node in the oStream tree, it will be lost at all downstream nodes. In VMesh, when a node consumes its current segment, it gets its next segment for playing from a new parent node by searching its next-segment-list or DHT search, so the node's suppliers and route

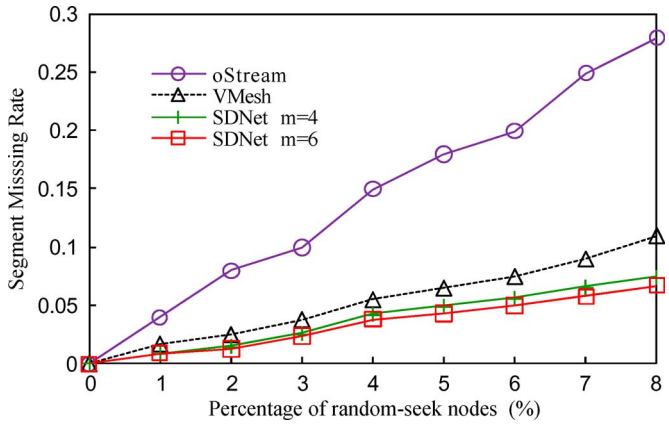


Fig. 9. The impact of random-seek rate.

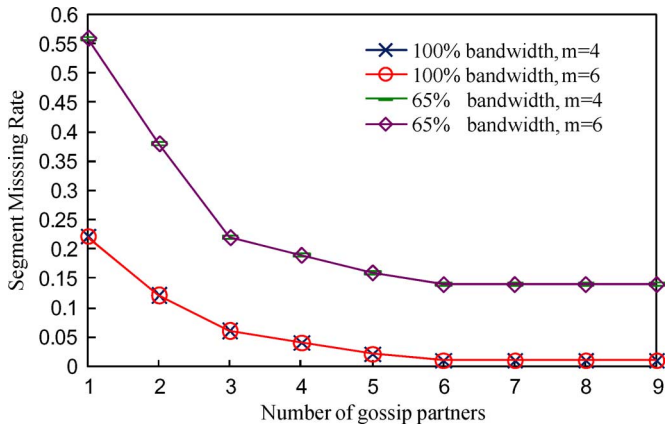


Fig. 10. The impact of the size of gossip partners and the dynamic network.

path change frequently with the playback time axis. However, in SDNet, the multi-way tree assists locating target node efficiently and when a node has established its gossip partners, the partnership is relatively stable, the multiple partners will continuously provide the streaming, which achieves better stability and adaptivity to dynamic network conditions and random-seek than both VMesh and oStream. From a video decoding point of view, the low and stable SMR achieves higher streaming quality.

4) Impact of Parameters Setting: Lastly the impact of parameter settings is considered. This paper focuses on two important parameters: the size of partners list and the playback buffer size of the node. Fig. 10 shows the results that under different network bandwidth and different size of partners list, the SMR decreases by the increasing the size of partners list because there are more nodes to supply the streaming, however, when the size of partners list is greater than 6, the decreasing of SMR is not obvious. Therefore setting partner list size to 6 is a reasonable choice.

Fig. 11 shows the impact on video quality under different playback buffer sizes. When system's scale is very small, the SMR is very high, because the number of nodes in system is small, so it is more difficult to find the streaming suppliers. At that time, an increase in the playback buffer size can significantly decrease the SMR. With the increasing of the system scale, SMR decreases and the impact of playback buffer size on

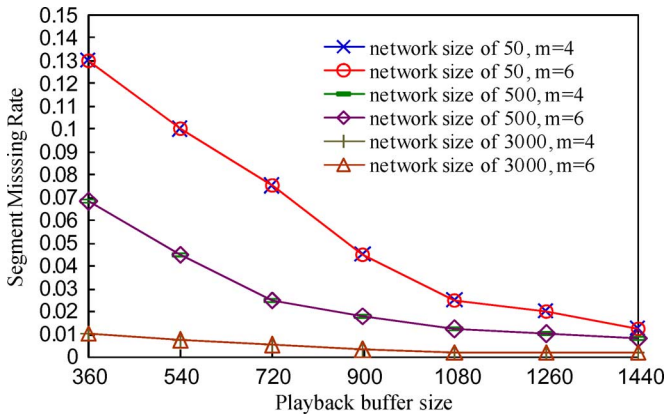


Fig. 11. The impact of playback buffer size under different system size.

SMR decreases. This is as when the system is big enough, it is easier to find appropriate suppliers.

C. Overhead Analysis

Firstly, the overhead of overlay construction and maintenance in SDNet is considered. Because SDNet integrates two networks: BATON* and DONet, each node needs to maintain parent-child and adjacent relationship in the tree and the gossip partnership in the DONet. Consequently the control overhead of SDNet is resulted from the addition of the control overhead of the two networks. BATON* shows how the cost of updating routing tables is $O(m\log_m N)$ for node join and departure. As it employs a light-weight gossip protocol, most control messages in DONet are for exchanging data availability information. The number of partners thus becomes a key factor to the control overhead. It is shown in [1] that as compared to video traffic, the control traffic is essentially minor. When VCR-like operations occur, the new architecture SDNet will introduce a set of new messages, one of them is the query cost for searching the new node corresponding to the new playing position in the tree. As it has been discussed above, the cost is $O(\log_m(m-1)\lceil Len/d\rceil)$.

Another issue is the cost of searching the new gossip partners in DONet, however, the numbers of gossip messages can be constrained by the maximal number of gossip partners and the value of $time\_to\_live$, which are independent of the size of network. Compared with the high-bandwidth streaming traffic, this overhead is negligible.

Secondly, the source media server stress is investigated. The server needs to provide extra bandwidth for pre-fetching video segments. In general case, $\lceil Len/d\rceil$ pre-fetching streams need downloading from source media server. If the total nodes are more than $\lceil Len/d\rceil$, the newly joined node can pre-fetch segments from its adjacent nodes (either the parent or sibling).

Following the discussion above, for SDNet, although there are extra load and bandwidth in comparison with any independent system, the overhead is limited and light-weighted and is balanced by the high quality of VoD services.

VII. Conclusion

This paper proposes SDNet, which enhances the existing Data-driven Overlay Network (DONet) solution to support

peer-to-peer VoD services. SDNet adds an assistant multi-way tree structure to the classic architecture in order to support high search efficiency associated with tree-based searches. Video segments are stored in the tree in a distributed manner based on the pre-fetching scheme. The cooperation between the novel multi-way tree with DONet makes SDNet to support VoD operations efficiently. The performance of SDNet has been extensively evaluated under various configurations. The results demonstrated that SDNet achieves low latency and low failure rate for VCR-like operations. SDNet also showed lower and stable video quality as estimated by a segment missing rate-based metric under highly dynamic network environments.

Future work aims at comparing SDNet solution with other proposed mechanisms including a version of VMesh (LA-POP) [8] which considers different popularity of exchanged content and SmartPIN [31] which focuses on both delivery performance and user-perceived quality.

REFERENCES

[1] X. Y. Zhang, J. C. Liu, B. Li, and T. S. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming," in *Proc. IEEE INFOCOM*, Miami, USA, March 2005.

[2] X. F. Liao, H. Jin, and Y. H. Liu, "AnySee: Peer-to-peer live streaming," in *Proc. IEEE INFOCOM*, Barcelona, Spain, April 2006.

[3] K. Sripanidkulchai, A. Ganjam, and B. Maggs, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points," in *Proc. ACM SIGCOMM*, Portland, OR, USA, August 2004.

[4] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: Peer-to-peer media streaming using CollectCast," in *Proc. ACM Multimedia*, Berkeley, California, November 2003.

[5] Y. Cui, B. Li, and K. Nahrstedt, "oStream: Asynchronous streaming multicast in application-layer overlay networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 91–106, Jan. 2004.

[6] C. X. Zheng, G. B. Shen, and S. P. Li, "Distributed pre-fetching scheme for random seek support in peer-to-peer streaming applications," in *Proc. ACM workshop on Advances in Peer-to-peer Multimedia Streaming (P2PMMS)*, Hilton, Singapore, Nov. 2005.

[7] W.-P. K. Yiu, X. Jin, and S.-H. G. Chan, "Distributed storage to support user interactivity in peer-to-peer video streaming," in *Proc. IEEE International Conf. on Communications (ICC)*, Istanbul, Turkey, June 2006.

[8] W.-P. K. Yiu, X. Jin, and S.-H. G. Chan, "VMesh: Distributed segment storage for peer-to-peer interactive video streaming," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1717–1731, December 2007.

[9] B. Cheng, H. Jin, and X. F. Liao, "Supporting VCR functions in P2P VoD services using ring-assisted overlays," in *Proc. IEEE International Conf. on Communications (ICC)*, Glasgow, Scotland, June 2007.

[10] C. Xu, E. Fallon, P. Jacob, and A. Hanley, "Distributed and tree-based pre-fetching scheme for random seek support in P2P streaming," in *The 7th Annual Information Technology and Telecommunications Conf. (IT&T)*, Dublin, Ireland, Oct. 2007.

[11] C. Xu, G.-M. Muntean, E. Fallon, and A. Hanley, "A balanced tree-based strategy for unstructured media distribution in P2P networks," in *Proc. IEEE International Conf. on Communications (ICC)*, Beijing, China, May 2008.

[12] C. Huang, J. Li, and K. W. Ross, "Can internet video-on-demand be profitable?," in *Proc. ACM SIGCOMM*, Kyoto, Japan, August 2007.

[13] T. Do, K. A. Hua, and M. Tantaoui, "P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment," in *Proc. IEEE International Conf. on Communications (ICC)*, Paris, France, June 2004.

[14] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: Peer-to-peer patching scheme for VoD service," in *Proc. International World Wide Web Con. (WWW)*, Budapest, Hungary, May 2003.

[15] S. Jin and A. Bestavros, "A cache-and-relay streaming media delivery for asynchronous clients," in *Proc. Int. Workshop on Networked Group Communication (NGC)*, Boston, Massachusetts, USA, October 2002.

[16] A. Sharma, A. Bestavros, and I. Matta, "dPAM: A distributed pre-fetching protocol for scalable asynchronous multicast in P2P systems," in *Proc. IEEE INFOCOM*, Miami, USA, March 2005.

[17] Y. Shan and S. Kalyanaraman, "Hybrid video downloading/streaming over peer-to-peer networks," in *Proc. IEEE International Conf. on Multimedia & Expo (ICME)*, Baltimore, Maryland, July 2003.

[18] H. V. Jagadish, B. C. Ooi, and K. L. Tan, "Speeding up search in peer-to-peer networks with a multi-way tree structure," in *Proc. ACM SIGMOD*. Chicago, Illinois, USA: , June 2006.

[19] F. Allamandri, S. Campion, and A. Centonza *et al.*, "Service platform for converged interactive broadband broadcast and cellular wireless," *IEEE Trans. Broadcasting*, vol. 53, no. 1, pp. 200–211, March 2007.

[20] W.-F. Poon, K.-T. Lo, and J. Feng, "Interactive broadcasting system for VBR encoded videos," *IEEE Trans. Broadcasting*, vol. 53, no. 2, pp. 459–467, June 2007.

[21] J. Padhye and J. Kurose, "An empirical study of client interactions with a continuous-media courseware server," in *Proc. the ACM Int'l Workshop on Network and Operating Systems Support for Design Audio and Video(NOSSDAV)*, Cambridge, UK, 1998.

[22] M. Chesire, A. Wolman, and G. Voelker, "Measurement and analysis of a streaming media workload," in *Proc. the USENIX Symp. Internet Technologies and Systems (USITS)*, Boston, Massachusetts, May 200l.

[23] A. Bharambe, C. Herley, and V. Padmanabhan, "Analyzing and improving a BitTorrent network's performance mechanisms," in *Proc. IEEE INFOCOM*, Barcelona, Spain, April 2006.

[24] D. L. Guan and S. Y. Yu, "A two-level patching scheme for video-on-demand delivery," *IEEE Trans. Broadcasting*, vol. 50, no. 1, pp. 11–15, March 2004.

[25] W. K. S. Tang, E. W. M. Wong, S. Chan, and K.-T. Ko, "Optimal video placement scheme for batching VoD services," *IEEE Trans. Broadcasting*, vol. 50, no. 1, pp. 16–25, March 2004.

[26] S. Chand and H. Om, "Modified staircase data broadcasting scheme for popular videos," *IEEE Trans. Broadcasting*, vol. 48, no. 1, pp. 274–280, Dec. 2002.

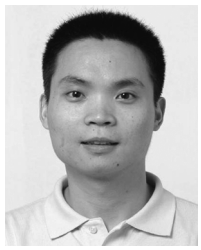
[27] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Networking*, vol. 11, no. 1, pp. 17–32, Feb. 2003.

[28] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie, "From epidemics to distributed computing," *IEEE Computer*, vol. 37, no. 5, pp. 60–67, 2004.

[29] "The Chord/DHash project," [Online]. Available: http://pdos.csail.mit.edu/chord/#downloads

[30] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOMM*, San Francisco, CA, USA, March 1996.

[31] S.-B. Lee, G.-M. Muntean, and A. Smeaton, "User-centric utility-based data replication in heterogeneous networks," in *IEEE International Conf. on Communications (ICC), Digital Television and Mobile Multimedia Broadcasting Workshop*, Beijing, China, May 2008.



Changqiao Xu received the B.Sc. and M.Sc. degrees in Software Engineering from Jiangxi Normal University, China in 1999 and 2002 respectively. He joined Institute of Software, Chinese Academy of Sciences (ISCAS) in 2002 where he held roles as a software engineer and project manager in the R&D area of network, multimedia transmission protocols and products. He joined Software Research Centre, Athlone Institute of Technology for his Ph.D. research work in 2007. His research interests include peer-to-peer multimedia streaming, innovative transport layer protocols, and multimedia transmission over wireless networks.

Gabriel-Miro Muntean is a Lecturer in the School of Electronic Engineering and co-Director of the Performance Engineering Laboratory at Dublin City University, Ireland, where he also obtained his Ph.D. degree in 2003 for research in quality-oriented adaptive multimedia streaming over wired networks. He was awarded the B.Eng. and M.Sc. degrees in Software Engineering from the Computer Science Department, "Politehnica" University of Timisoara, Romania in 1996 and 1997 respectively. Dr. Muntean's research interests include QoS and performance issues of adaptive multimedia streaming, and personalized eLearning over wired and wireless networks and with various devices. Dr. Muntean has published over 70 papers in top-level international conferences and journals, as well as an authored book, two edited books and four book chapters. Dr. Muntean is reviewer for important international journals, conferences and funding agencies. He is a member of IEEE, IEEE Broadcast Technology Society and an Associate Editor of the IEEE TRANSACTIONS ON BROADCASTING.

Enda Fallon is manager in the Software Research Center (SRC), Athlone Institute of Technology, Ireland. He has combined a career in industry with academic research. In 2002 Enda joined Athlone Institute of Technology from Ericsson. As manager of the SRC he has been extremely successful in attracting collaborative industry/academic research funding. Since its inception the SRC has attracted 2M in collaborative research funding with industrial partners such as Ericsson, Eircom and Intelliden. He holds a BSc in Computer Science and Mathematics from University College Galway, an MSc in Software Engineering from Athlone Institute of Technology and is currently undertaking a PhD with the Performance Engineering Laboratory at University College Dublin, Ireland.

Austin Hanley is head of School of Engineering at Athlone Institute of Technology (AIT), Ireland since 2003. He joined Ericsson in 1980 and over a 20-year career, held software design, systems design and product management posts in Athlone, Dublin and Stockholm. He became General manager of Ericsson's International Training centre in Dublin, was appointed to lead up a large O&M development project for a new switching system at Ericsson research laboratories in Stockholm (Ellemtel) before becoming GM of Ericsson's 500-man R&D facility in Ireland. He set up his own company (Libertus Ltd) in 2001 and a strategic business group for Ireland's Industrial Development Agency (IDA) and consulted to a number of multi-national companies. He is the co-founder of the successful Software Research Centre at AIT. Austin holds a BA (Honors) degree from University College Galway, a H.Dip (Ed) from National University of Ireland Maynooth, H.Dip (Computer Science) and MSc (Computer Science) degrees from University of Limerick and a MSc (Management) degree from Trinity College Dublin (TCD). He is currently completing a doctorate at TCD. He holds two patents and has a number of published papers.