DONET-VOD: A HYBRID OVERLAY SOLUTION FOR EFFICIENT PEER-TO-PEER VIDEO ON DEMAND SERVICES

Changqiao Xu[1, 3, 4], *Gabriel-Miro Muntean*[2], *Enda Fallon*[1], *Xiaoguang Li*[1]

[1]Software Research Centre, Athlone Institute of Technology, Ireland
[2]School of Electronic Engineering, Dublin City University, Ireland
[3]Institute of Software, Chinese Academy of Sciences, China
[4]Graduate University of the Chinese Academy of Sciences, China

ABSTRACT

The existing DONet-based approach uses successfully a random gossip algorithm for scalable live video streaming. This pure mesh overlay network-based solution may lead to unacceptable latency or even failure of VCR operations in Video-on-Demand (VoD) services where nodes usually have different playing offsets, across a wide range. This paper proposes DONet-VoD which enhances DONet in order to address issues related to VoD delivery and VCR operations. In DONet-VoD, DONet principle is employed for the video distribution over the overlay network and a novel algorithm which uses a multi-way tree structure and extra prefetching buffers at the nodes is proposed to support efficient VoD operations. Video segments are prefetched and stored in a distributed manner in the nodes' prefetching buffer along the tree. 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 prefetching scheme, details the cooperation procedure, and then analyses the performance of the proposed DONet-VoD .

Index Terms—Peer-to-peer, VoD, prefetching, multi-way tree, DONet

1. INTRODUCTION

There is extensive recent research in the area of peer-to-peer (P2P) multimedia streaming [1-9]. Unlike P2P live streaming [1, 2] in which the peers start playback from the current point of streaming when they join the streaming session, in P2P Video-on-Demand (VoD) streaming [3-6] VCR-like operations such as forward, backward, and random-seek have to be supported. Providing this level of interactive streaming service in a P2P environment is a significant challenge.

"Cache-and-relay" mechanism [7] is used in P2P streaming which caches the played content and relays it to the peers which request streaming of the same content. Furthermore, the mechanism is extended [8] by caching apart from the played portion of the video stream also extra sequences prefetched in advance by using additional bandwidth. The prefetching strategy can grant peers the ability to overcome the bursty packet loss, the departure of source peer, and to smoothen the playing experience.

Most research use prefetching strategy to assist solving the interactivity problem for multimedia streaming. Researchers proposed a hierarchical prefetching scheme for popular or sub-popular segments [4], which is based on the assumption that the popular or sub-popular segments will be visited often with the random-seek. However, this scheme's main challenge is collecting the user viewing logs in a distributed P2P system. In VMesh [5], video segments are prefetched and stored in nodes over a Chord network. Each node maintains previous/next-segments-lists based on distributed hash tables (DHT) and uses the lists to support jumps. VMesh does not use "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.

DONet [1] uses successfully a random gossip algorithm for scalable live video streaming, but this mesh overlay network-based approach may lead to unacceptable latency for VCR operations in VoD services where nodes usually have different playing offsets. In RINDY [6], 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 (e.g. following a random-seek). Though RINDY decreases the latency for locating target node when random-seek occurs compared with DONet, it has the same problems as DONet: increase of both jump latency and failure rates. With both RINDY and DONet, 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.

This paper proposes DONet-VoD, which enhances DONet with a novel algorithm to address VoD VCR operations-related issues and to increase their efficiency. DONet-VoD uses a multi-way tree structure of additional prefetching buffers at the nodes. Video segments are prefetched and stored in the nodes' prefetching buffers along the tree in a distributed manner. The paper presents and discusses DONet-VoD's algorithm and analyses the performance of the proposed solution.

2. DONET-VOD ARCHITECTURE AND PRINCIPLE

DONet-VoD has a hybrid architecture (see figure 1), which integrates two networks: a multi-way tree and DONet. The multi-way tree is based on BATON* [9] which is balanced. The cost of the search in BATON* is bounded by $O(log_m N)$.

Apart from the playback buffer used during video streaming, DONet-VoD involves the addition of an extra prefetching 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 prefetching buffer. The video segments are not selected at random; they are chosen by the prefetching scheme, which orders the video segments in the adjacent nodes' prefetching buffers according to the playback time. After the video segments are completely downloaded, they will be maintained during the node's lifetime.

In DONet-VoD, DONet is the routine overlay for video distribution. Under normal conditions, each node gets the multimedia streaming from its gossip partners. When VCR-like operations occur, the partners updating progress is triggered 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 prefetching buffers. After re-establishing new partners by using *REQUEST* and *REPLY* messages (detailed in section 4) over DONet, the streaming suppliers are switched to the new partners again.

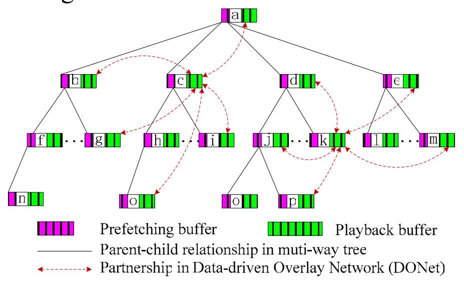
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
o
p
Prefetching buffer
Playback buffer
Parent-child relationship in muti-way tree
Partnership in Data-driven Overlay Network (DONet)

Fig.1 The architecture of DONet-VoD

3. DONET-VOD PREFETCHING SCHEME

Assuming that the notations presented in table 1 are used, if the video *P* is coded using CBR rate and is divided into equal segments (e.g. one segment is 1 sec. playback length),

TABLE I
NOTATIONS USED OF PREFETCHING ALGORITHM

Notations	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 prefetching 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 prefetching buffer
Seg(*X*)	The sequence number of prefetching unit in *Prebuf*(*X*)
Parent(*X*)	Node *X*'s parent node in *T*

d numbers of successive segments are set as one prefetching unit, then *P* has $L = \lceil Len/d \rceil$ prefetching units which are numbered from 1 to *L* sequentially. The prefetching 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 prefetching scheme is illustrated in figure 2. As it shows, at the level 0 of *T*, we set the mapping relationship between the corresponding prefetching unit of *P*'s middle position with *Prebuf*(*R*), namely $Seg(R) = \lceil L/2 \rceil$. At level 1 the children nodes (from left to right) of *R* are named R_1, R_2, …, R_m and *P* is divided into *m* equal subsections $L_1, L_2, \ldots, L_m$. Assuming $Mid(L_1)$, $Mid(L_2)$, … , $Mid(L_m)$ are the prefetching units' sequence numbers corresponding to the middle position of L_1, L_2, …, L_m respectively, we set $Seg(R_1)=Mid(L_1)$, $Seg(R_2)= Mid(L_2)$, … , $Seg(R_m)= Mid(L_m)$. 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 prefetching unit). Then, we set the prefetching unit's sequence number to equal its parent node's. For node *X*, presuming *X* is the i^{th} child of *Parent*(*X*), *Level*(*R*)=0. From the settings and notations presented above, the following equations can be deduced:

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

$$Seg(X) = \lceil L/2 \rceil \qquad (1)$$

(2)If $0 < Level(X) \le \lfloor \log_m (m-1)L \rfloor$, there are two cases:

① If $1 \le i \le \lceil m/2 \rceil$, then

$$Seg(X) = Seg(Parent(X)) - \left\lceil \frac{(m-2i+1)L}{2m^{Level(X)}} \right\rceil \qquad (2)$$

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

$$Seg(X) = Seg(Parent(X)) + \left\lceil \frac{(2i-m-1)L}{2m^{Level(X)}} \right\rceil \qquad (3)$$

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

$$Seg(X) = Seg(Parent(X)) \qquad (4)$$

For conditions (1), (2), (3), X needs prefetching of the specific prefetching unit from S; for condition (4), X gets the prefetching unit from its *Parent*(X) or one of its adjacent nodes whose prefetching unit sequence number equals *Seg*(X). (4) avoids to download all the prefetching units from S, alleviating the load of S.

BATON* is adaptive to dynamic network, for node join, departure, failure or load balancing. If a node's position in T is changed, it should check whether the prefetching unit in its prefetching buffer satisfies the equations (1)-(4); if not it should update the prefetching unit.

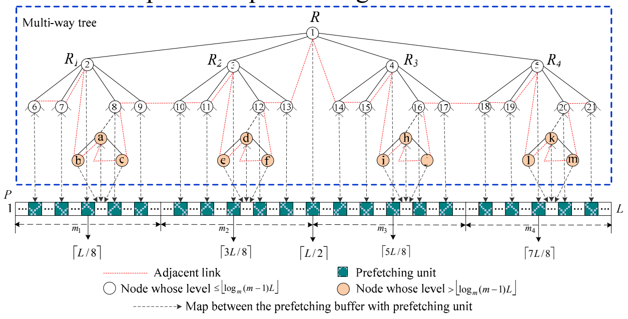

Fig.2 Prefetching scheme

4. SUPPORTED VOD SERVICES

We design two types of messages *REQUEST* and *REPLY* to locate new partner required by jump operation in DONet for the partners updating progress. The message is <*typeid*, *segid*, *srcid*, *TTL*, *routepath*, *resultset*>, *typeid* represents the type of message (*REQUEST* or *REPLY*); *segid* is the target segment sequence number corresponding to the new jump position; *srcid* denotes the source address of the message; *TTL* is the remaining hop count of this message; *routepath* is a list of peers traversed by the request message; *resultset* stores the result of query. Assuming X triggers partners updating progress; 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 *segid* or *TTL* becomes zero. The final 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 all partners of *resultset* into its partners list. If X can not find the target node until *TTL*=0, X sends *REQUEST* to another node in its *mCache* and try again.

When a node joins system, it implements two node join progresses for BATON* and DONet respectively. When the node starts viewing the video, it implements prefetching scheme using its residual bandwidth. During the playback, assuming X jumps another video position, the follows operations are performed: (1) *segid* is calculated approximately from the player interface, then the corresponding prefetching unit is $\lceil segid/d \rceil$; (2) X empties its partnership list, partners updating progress is triggered over the DONet; (3) The tree is traversed until node J with *Seg*(J)= $\lceil segid/d \rceil$, If J cannot be found until leaf node is visited, failure is returned; (4) Search the right adjacent link of J, assuming *Que* is a queue with size of k, J is put into *Que*. *Ln* is the tail of *Que*, *Cn* is the present visited node traversing the adjacent link. If *Seg*(*Cn*) = *Seg*(*Ln*)+1, *Cn* is put into *Que*. If *Seg*(*Cn*) = *Seg*(*Ln*) and the present usable bandwidth of *Cn* is more than *Ln*'*s*, then *Ln* is replaced by *Cn*. If *k'*(default=5) successive visited nodes have the same prefetching unit, then switch to R directly, search from R to locate the node whose prefetching unit equals *Seg*(*Ln*)+1, then continues search its right adjacent node. The search progress repeats until finding the node whose prefetching unit equals *Seg*(*Ln*)+1 fails or *Que* is full; (5)The nodes in *Que* send video segments in their prefetching buffer to X; (6) When X finishes re-establishing new partners, *Que* is notified to stop sending streaming. Then, the multimedia streaming suppliers are switched to X's new partners. Otherwise, new visited nodes are put into *Que* if all the nodes in *Que* have finished sending prefetching streaming.

As fig. 3 shows, node 3 jumps, the target segment corresponding to the new position is g. After search in T, g falls into node 4's prefetching buffer. During 3 updating its partners list, nodes 4, o, 16, 17, 18 send its prefetching buffer's video segments to 3 until 3 has established new partners which are 6, h, k, 13, then the new partners will supply the streaming.

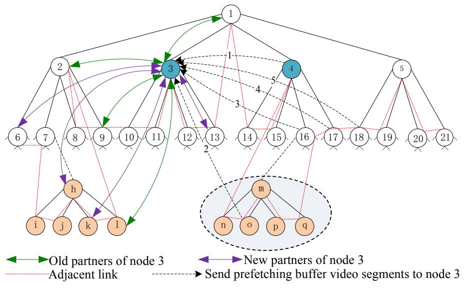

Fig. 3 Collaboration of tree with DONet

5. PERFORMANCE EVALUATION

In this section, we analyze the performance of DONet-VoD. Assuming the fanout of the multi-way tree is m, w is the playback buffer window size of the peers.

Jump Latency: When jump occurs, the prefetching buffer will provide the streaming service temporarily before the node finishes re-establishing its partners. Consequently the jump latency is in terms of hop count for locating the node in the tree; the segment corresponding to the new position falls into this node. The node's right adjacent nodes will constitute the successive streaming for the new scene using their prefetching buffers. Taking the features of BATON* and the prefetching scheme into account, the hops for search are less than $O(\log_m (m-1) \lceil Len/d \rceil)$. This results shows how DONet-VoD is more efficient than VMesh and RINDY. In VMesh, the cost for search via DHT is $O(\log N)$. In RINDY, the cost for search between rings is $O(\log\lceil Len/w \rceil)$. Assuming Len=7200 (two hours for playback), w=10, d=10. Figure 4 presents a result comparison when different parameters are used.

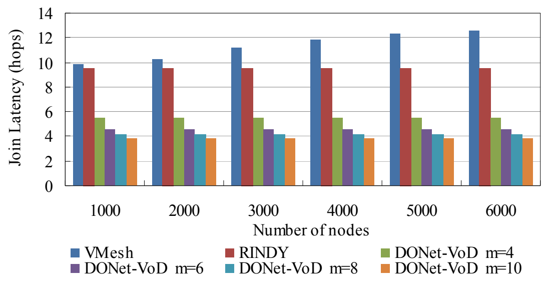

Fig.4 Jump latency

Jump Failure: The number of joined nodes in DONet-VoD impacts the average jump failure rate. In general case, if the total joined nodes are more than $\lceil Len/d \rceil$ at any time, the jump operation will be successful, as prefetching buffers of all the nodes have shared the whole video segments of the requested video. However, in RINDY, the average jump failure rate does not only relate with the number of joined nodes, but also it is impacted by other nodes' present playing position. If the nodes are close on the playback time axis, or the target node also initiates jump, the operation of jump fails.

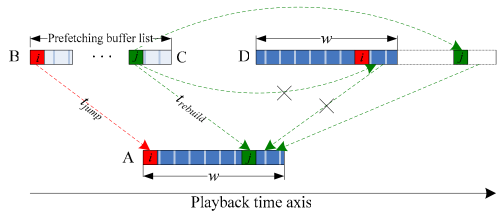

Fig. 5 The impact of playback window size

In DONet-VoD, though the unchanged prefetching buffer ensures the high success rate for jump, the video segments in prefetching buffer are provisional suppliers, it is not good that one node gets the streaming from other nodes' prefetching buffer for a long time, because they also need to assist other nodes for jump. So the node needs to establish the new partners as soon as possible. As figure 5 shows, assuming node A initiates jump at time t_{jump}, the segment of the new position is i. During A updating partners, nodes B, C provide streaming for A using their prefetching buffer. At time $t_{rebuild}$, A finishes updating, D is its new partner which includes i. Assuming j is the present playing segment in A, j should be around $i + t_{rebuild} - t_{jump}$, but it is possible that j falls out D's playback buffer, A finding partner D fails actually, if w is strengthened, D can become A's partner, so we need set appropriate size of w to increase the probability for search partners in the collaboration between the two networks.

Overhead: Compared with DONet, DONet-VoD needs to maintain another network: the multi-way tree. BATON* shows how the cost of updating routing tables is $O(m\log_m N)$ for node join and departure. Considering the source media server stress, it needs to provide extra bandwidth for prefetching video segments. In general case, $\lceil Len/d \rceil$ prefetching streams need downloading from S. If the total nodes are more than $\lceil Len/d \rceil$, the new joined node can prefetch segments from its adjacent nodes, either the parent or sibling. Though extra loads increase, they are light-weighted and will not bring too much overhead to the DONet. The cost of extra bandwidth and control overhead are balanced by the high quality of VoD services.

6. CONCLUSIONS

This paper proposes DONet-VoD, which enhances existing DONet solution to support peer-to-peer VoD services. DONet-VoD adds an assistant multi-way tree structure to the classic architecture in order to support high search efficiency associated with the tree searches. Video segments are stored in the tree in a distributed manner based on the prefetching scheme. The cooperation between the novel multi-way tree with DONet makes DONet-VoD to support VoD operations efficiently.

7. REFERENCES

[1] X.Y. Zhang, J.C. Liu, and B. Li, "CoolStreaming/ DONet: a data-driven overlay network for peer-to-peer live media streaming," Proc. IEEE INFOCOM'05, March 2005.

[2] X.F. Liao, H. Jin, and Y.H. Liu, "AnySee: peer-to-peer live streaming," Proc. IEEE INFOCOM'06, April 2006.

[3] C.Q. Xu, G.-M. Muntean, and E. Fallon, "A balanced tree-based strategy for unstructured media distribution in P2P networks," Proc. IEEE ICC'08, May 2008.

[4] C.X. Zheng, G.B. Shen, and S.P. Li, "Distributed prefetching scheme for random seek support in peer-to-peer streaming applications," Proc. ACM workshop on Advances in peer-to-peer multimedia streaming (P2PMMS'05), Nov 2005.

[5] W.-P. Ken Yiu, X. Jin, and S.-H. Gary Chan, "Distributed storage to support user interactivity in peer-to-peer video streaming," Proc. IEEE ICC '06, June 2006.

[6] B. Cheng, H. Jin, and X.F. Liao, "Supporting VCR functions in p2p vod services using ring-assisted overlays," Proc. IEEE ICC'07, June 2007.

[7] S. Jin, A. Bestavros, "A Cache-and-relay streaming media delivery for asynchronous clients," Proc. Int. Workshop on Networked Group Communication (NGC'02), Boston, USA, October 2002.

[8] A. Sharma, A. Bestavros, and I. Matta, "dPAM: a distributed prefetching protocol for scalable asynchronous multicast in P2P Systems," Proc. IEEE INFOCOM'05, March 2005.

[9] H.V. Jagadish, B.C. Ooi, and K.L. Tan, "Speeding up search in peer-to-peer networks with a multi-way tree structure," Proc. ACM SIGMOD'06, June 2006.