Mediacoop: Hierarchical Lookup for P2P-VoD Services

Tieying Zhang
Institute of Computing Technology
Chinese Academy of Sciences
zhangtiey@software.ict.ac.cn

Jianming Lv
Department of Computer Science
South China University of Technology
jmlv@scut.edu.cn

Xueqi Cheng
Institute of Computing Technology
Chinese Academy of Sciences
cxq@ict.ac.cn

Abstract

The random seeking in P2P-VoD system requires efficient lookup for “good” suppliers. The main challenge is that good suppliers should meet two requirements: “content match” and “quality match”, while most existing methods only focus on one aspect. In this paper, we propose Mediacoop, a novel structured lookup method combining both content and quality match to provide random seeking for P2P-VoD services. It exploits playpoint distance to efficiently locate the candidate suppliers with required data (content match), and performs refined lookup within the candidates to meet quality match. Theoretical analysis and simulations show that Mediacoop outperforms the traditional methods. Our real-world system also proves the effectiveness of the design.

1. Introduction

With the rapid deployment of broadband access into household, P2P-VoD services have become one of the most popular Internet applications due to the free user interactivity (e.g., random seeking and access) [2]. However, in P2P-VoD systems, requester fetches data directly from the streaming buffer of suppliers, and the random seeking operations often make the current supplying peers useless. Then, the goal is to efficiently find suppliers whose buffer stores the required block (content match) with good quality (quality match).

The goal involves two aspects. One is “content match”. That means the supplying peers we found can provide the required block, which is the baseline for the lookup result. The second one is “quality match”, which means the supplying peers should be of good physical performance, such as high upload bandwidth and less physical delay with the requester. Quality match is not only an optimization but also a critical factor for on-demand streaming services, as supplying peers are always unable to provide enough data to satisfy the request due to the physical quality [2, 3, 7]. Two important metrics for measuring physical quality are delay and bandwidth [11]. In this paper, we use delay as the quality performance, and the reasons will be given in Section 2 and 4.1.

Until recently, most existing methods only focus on one aspect of the goal for P2P-VoD lookup. A typical approach is using distributed hash table (DHT) based network (e.g., [6, 7]). It publishes the information of streaming buffer to DHT. However, the content of the buffer change constantly as playing, and peer has to continuously update the DHT accordingly, which results in huge overhead. Although the method in [7] also explores peer quality, it is not aiming at random seeking, and can not handle the interactive operations. Work [8, 9] propose non-DHT structured methods to avoid huge publishing overhead. However, they do not exploit the physical quality of peers.

In this paper, we propose Mediacoop, a hierarchical structured lookup method combining both content and quality match to provide random seeking and access for P2P-VoD. The lookup process in Mediacoop is divided into two stages. In the first stage, we use unchanged playpoint distance to locate the candidate suppliers with the required data block. In the second stage, we index the candidates into a novel tree-like sub-overlay. Then, refined search is performed along the sub-overlay to efficiently find “good” suppliers.

Our contribution can be summarized as follows:

1. We propose an efficient structured lookup method combining both content and quality match function to provide random seeking and access for P2P-VoD. The lookup process in Mediacoop is divided into two stages. In the first stage, we use unchanged playpoint distance to locate the candidate suppliers with the required data block. In the second stage, we index the candidates into a novel tree-like sub-overlay. Then, refined search is performed along the sub-overlay to efficiently find “good” suppliers.

2. To meet quality match, we design a tree-like structure to index candidate suppliers as a sub-overlay, which provides efficient quality match lookup.

3. We have performed theoretical analysis and extensive simulation experiments to compare with the
traditional methods. The results show that Mediacoop can significantly reduce startup and seeking latency while improving continuity closer to 1.0 with less overhead. We have also implemented a real-world P2P VoD system over the Internet, called CoolFish [1]. The running results also prove our design.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 describes the model of our idea. Section 4 explains the design in detail. In Section 5, we evaluate the performance of Mediacoop by theoretical analysis and show the results in simulations and real-world system. In Section 6, we conclude the paper with the discussion of future work.

2. Related Work

Tree-based network has been proposed to provide VoD services. P2Cast [4] uses patching techniques to stream video, while relying on unicast connections among peers. However, the single parent delivery model is not efficient enough in a heterogeneous network. Moreover, it is not easy to maintain a tree structure in a dynamic environment. P2VoD [5] organizes each video session tree into layers. Peer departure is handled by finding another parent in the upper layer and new client can join the lowest layer of the tree or creating a new layer. However, P2VoD does not provide any mechanism for peer to random seek.

Recently, researchers use mesh-based network as improvement of tree-based overlay. PROP [6] is built on top of DHT-based network to provide VoD service. In PROP, when a peer fetches a video block, it publishes the information to DHT. If another peer wants to fetch a block, it inquires the DHT and finds a supplier. A problem of this approach is that when a peer moves on and discards some blocks from its buffer, it needs to send deleting messages to update the DHT. PROMISE [7] exploits network bandwidth to provide peer selection. It uses the actual media data to probe available bandwidth during the peer’s playing, where probe process is performed after the peer’s access or seeking. Therefore it can not handle the random seeking in VoD, and this is why the method is called “selection” rather than “lookup”. More relevant to our work are OBN [8] and RANDY [9]. OBN considers the DHT update problem, where it uses the buffer relationship between peers to construct a non-DHT structured overlay. However, OBN does not exploit the physical quality of peers. RANDY uses a similar approach to construct a multi-ring lookup network. Requester firstly finds the ring containing the target peers. Then these peers can be located through gossip protocol. However, using gossip protocol is costly when the ring contains a large number of peers and it also does not consider the peer quality.

3. System Model

Two key problems need to be solved in Mediacoop. One is how to provide content match lookup efficiently; the other is how to find close peers with less physical delay. To address these two issues, our search process is divided into two stages along hierarchical overlays.

3.1. Playpoint Overlay

The content of video is segmented into $M$ blocks and each block corresponds to a playpoint. We group peers with the same playpoint into one playpoint (see Fig. 1(a)). In view of the fact that the distance of playpoints is unchanged due to the constant playback rate for a given movie, our idea in the first stage is to utilize the distance of playpoints to index all swarms on a ring (Fig. 1(b)). Therefore every swarm is acquainted with the status of each other even though they are moving. Requester can easily find peers according to a Chord-like structured overlay [13]. The difference is that we use the distance of playpoint instead of hash value, and the unchanged distance leads to no requirement for update messages, which reduces huge overhead.
3.2. Sub-Overlay: Indexed Swarm

The result of the first stage is a peer (we call this peer seed) belonging to some swarm. Actually, all of the peers in the swarm meet the requirement of content match. Our goal is to find some peers who have less physical delay with the requester (quality match). Inspired by current physical delay detection approach which can get the AS level or IP prefix level delay table [10], our idea in the second stage is to index the candidate peers using their IP prefixes as a sub-overlay, then we can find the target IP prefix peers along the sub-overlay assisted by delay table. The details how to get the delay table without ISPs’ help will be explained in Section 4.1.

4. Design of Mediacoop

4.1. Exploit Physical Quality

As an interactive application, end-to-end delay is important [10] for P2P system. Especially in VoD services, the user interactivity frequently happens [2], large delay between peers results in long response time. The International Telecommunication Union G.114 [12] recommends 150 ms as the upper limit for one-way delay for most interactive applications. Therefore, end-to-end delay is adopted to measure the physical ability of supplying peer and 150 ms is the upper bound. Although end-to-end delay varies with time, we are not interested in the exact value. We only need to record the average delays for different candidates, and separate the smaller ones as suppliers. The question is how to get the end-to-end delay. The Internet is composed of many Autonomous Systems (ASes). Peers within one AS are usually close to each other and inter-AS routing is specified by Border Gateway Protocol (BGP). Thus, we only need to know the AS-AS delays or cluster level delays and choose supplying peers from those clusters whose delay is the least with the initiator. We use the method proposed in [10] to construct the AS topology and obtain AS-AS delays. A sketch of the method is briefly described as follows (cf. [10] for more details).

Firstly, we collect a large number of public BGP routing tables and BGP updates, such as those from RouteViews (http://www.routeviews.org) and RIPERIS (http://www.ripe.net/projects/ris). From these routing tables, we build an AS graph and group IPs with the same longest prefix into one cluster. Then, we randomly choose one IP out of each cluster as the cluster delegate. A delay measurement between each pair of cluster delegates can be estimated by the tool King [14]. Finally, we obtain two tables: (1) IP to cluster mapping table (ICMT); (2) Cluster to cluster delay table (CCDT). From ICMT, a peer can learn about its own cluster, and it can get the target clusters from CCDT, who has the least delay with itself. Fig. 2 illustrates the work procedure of delay detector, and Table 1 shows a CCDT example in CoolFish. Now, the key problem is how to organize these clusters to provide efficient search.

In our approach, clusters are organized into a binary tree, among which every leaf node represents a cluster. Therefore, the number of the leaf nodes K is equal to that of clusters. Every leaf node is annotated with the physical address (IP prefix) which is represented by prefix code. The distance between leaf node1 and node2 is calculated by XOR of their prefix codes. In formulation, the distance is defined as follows:

\[ D_{\text{ip-prefix}}(n_1, n_2) = \text{Prefix}_{n_1} \oplus \text{Prefix}_{n_2} \]  

Note that, the candidate peers distribute among the clusters (leaf nodes) and empty clusters cannot be indexed (see in Fig. 3(b)). Some might argue that why we use tree structure to organize IP prefixes. The reason is that distance between leaf nodes in prefix binary tree is consistent with the measurement of XOR operation.

4.2. Peer State

Mediacoop performs lookup along hierarchical overlays. Accordingly, every peer has two finger tables.

Playpoint finger table (PPFT). PPFT stores playpoint information about other swarms. It has \( \log M \) (M is the number of blocks) items. For each \( 0 \leq i < \log M \), it keeps information for \( k \) peers of playpoint (playblock) distance between \( 2^i \) and \( 2^{i+1} \) from itself. The distance \( p_1 \rightarrow p_2 \) is defined as the number of hops from \( p_1 \) to \( p_2 \) in clockwise along the ring. It can be denoted as:

\[ D_{\text{playpoint}}(p_1, p_2) = \left( \left( PP_{p_2} \text{ Mod } M \right) - PP_{p_1} \right) \text{ Mod } M \]  

where \( M \) is the number of blocks, \( PP_{p_0} \) is \( p_0 \)'s playpoint.

Fig. 4(a) presents the structure of PPFT. Every item records \( k=2 \) peers whose PP is within the distance interval \( [2^i, 2^{i+1}) \). If the number of peers in the interval is larger than \( k=2 \), we only record 2 peers’ information according to LRU.
IP prefix finger table (IPFT). IPFT records IP prefixes of other peers within the same swarm. IPFT has $\log K$ ($K$ is the number of leaf nodes) items. For each $0 \leq j < \log K$, it keeps information for the peers of distance between $2^j$ and $2^{j+1}$ from itself. The distance is calculated according to (1). Fig. 4(b) shows the IPFT, where the structure is similar to PPFT.

<table>
<thead>
<tr>
<th>Playpoint Distance</th>
<th>Routing Peer</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^0$</td>
<td>$P_0$ (IP:Port) $P_1$ (IP:Port)</td>
</tr>
<tr>
<td>$2^1$</td>
<td>......</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
</tbody>
</table>

**Fig. 4. Finger table.**

**4.3. Lookup along Hierarchical Overlays**

**Content match lookup.** This is the first stage lookup along playpoint overlay. A recursive algorithm is employed for the lookup. On receiving lookup message, recipient firstly calculates the distance $D$ from itself to the required block according to (2). Then, the recipient picks the closest peer from its PPFT and relay the message to it. This algorithm terminates when the closest peer’s distance is equal to $D$ or there is no distance closer than $D$. In the former case, the closest peer is returned while in the later case, the recipient itself is returned.

**Quality match lookup along candidate swarm.** After the initiator $P$ obtains the seed from the first stage, it gets $\alpha$ target IP prefixes which have less delay with itself in CCDT, where $\alpha$ is system parameter (in simulation, we set $\alpha=3$). Then $P$ sends IP prefix lookup messages to the seed. Seed relays the messages to the closest peers in its IPFT. The relay peers perform the same operation recursively until there are no route peers with closer distance than itself. The relay peer with IP address meeting the requirement will be returned. If there is no answer to the initiator, $P$ will ask seed randomly return several peers in its swarm, as these peers meet the content match requirement although they might have large end-to-end delays.

**An example.** Fig. 3 shows an example of the whole lookup process from $p_0$. In Fig. 3(a), $p_0$ initiates a lookup with the goal to find suppliers who can provide block8. It calculates $D_{playpoint}(p_0,block8) = 6$ and sends the “content match” lookup message to the closest neighbor $p_8$. In the same way, $p_8$ calculates $D=1$ and finds that $p_{13}$ belongs to the candidate swarm. Therefore, $p_{13}$ is returned to the initiator as a seed. Then $p_0$ looks up its CCDT, and picks out the target IP prefix 210.77.* which has the least delay with $p_0$. Then $p_0$ sends the IP prefix lookup message to $p_{13}$ to starts the second stage (see Fig. 3(b)). On receiving the message, $p_{13}$ looks up IPFT and relay it to $p_{14}$ which is the closet one. In the same way $p_{14}$ relays the massage to $p_{15}$ and finally finds the target peers within cluster 210.77.*.

**4.4. Maintenance**

To join the network, peer $P$ must contact an already existing peer $J$. $P$ inserts $J$’s information into its own
finger table. Then \( P \) performs a peer lookup for its own attributes (playpoint and IP prefix). After that, \( P \) gets the closest neighbor \( Y \) in the network. Finally, \( P \) refreshes its finger table further away than its closest neighbor \( Y \) with \( Y \)'s finger table. During the refreshes, \( P \) also inserts itself into others' finger tables.

When peer \( P \) performs VCR operations, such as pause, stop and seeking, \( P \) will notify its neighbors (in both two finger tables) its new playpoint and its neighbors will update the information.

In our system, the cluster-delay detector program is running on a probing server and updating ICMT and CCDT all the time. Note that, CCDT records the average delay values. When a peer joins the system, it gets ICMT and CCDT from the probing server and asks server for update every 30 minutes.

5. Performance evaluation

5.1. Theoretical Analysis

In this section, we analyze the efficiency for our two stages lookup approach. We can model the performance as follows:

\[
P(M, K) = P_{FirstStage}(M) + P_{SecondStage}(K)
\]

where \( P(M, K) \) is the total lookup hop counts of Mediacoop; \( P_{FirstStage}(M) \) and \( P_{SecondStage}(K) \) is the hop counts of the first and second stage respectively; \( M \) is the number of blocks and \( K \) is the number of clusters; First we analyze \( P_{FirstStage}(M) \) for the first stage.

Mediacoop is a structured method and the lookup procedure is similar to the DHT protocols, e.g., Chord. However, we use playpoint (playblock) instead of peer identifier in DHTs. Therefore the performance is not only the traditional \( O(\log N) \) in DHTs, but also is related to the number of playpoints (i.e., the number of blocks):

\[
P_{FirstStage}(M) = \min \{ O(\log M), O(\log N) \}
\]

where \( N \) is the total number of peers. In general, the number of peers for a popular P2P system is usually very large. In contrast, the number of blocks for a video is severely limited. For example, according to our experience, 720 blocks is enough for a common 2-hour movie. That is, \( M \ll N \), thus:

\[
P_{FirstStage}(M) = \min \{ O(\log M), O(\log N) \} = O(\log M)
\]

For the second stage, the lookup procedure is essentially a binary search along the search tree. Therefore, the lookup performance of the second stage is equal to the time-complexity of binary search:

\[
P_{SecondStage}(K) = O(\log K)
\]

where \( K \) is the number of valid IP clusters. Invalid IP clusters mean that they are empty with no peers within them and will not be taken into consideration. In fact, the second stage is conducted in the target swarm with the number of peers \( n = N/M \) (we can assume that all peers are uniformly distributed among playblocks). Thus, \( K \) is actually equal to or less than \( n \), where \( n \) is the maximum value of \( K \) when each cluster only has one peer. Therefore, the total complexity of our hierarchical lookup method is:

\[
P(M, K) = P_{FirstStage}(M) + P_{SecondStage}(K)
\]

\[
\leq O(\log M) + O(\log N / M) = O(\log N)
\]

that is:

\[
P(M, K) \leq O(\log N)
\]

That is to say, in equal to or less than \( O(\log N) \) hops we can find out the peers, which not only have the required content but also have strong ability to meet the delay requirement for interactive applications.

5.2. Simulation Metrics and Configurations

We evaluate our system using the following performance metrics:

1. **Average number of hops** - the average number of routing hops from the beginning to the termination of the lookup process.

2. **Server stress (Mbps)** – the outgoing bandwidth required at the media server to support the whole system. We use the peak stress to examine the system scalability.

3. **Playback continuity (0-1) -** the number of pieces that arrive before or on playback deadlines over the total number pieces. The higher the continuity index, the better the video quality is.

4. **Startup latency (second)** - the time period starting from the moment a peer joins the network to the moment it starts playing the required video after buffering 5 seconds video data.

5. **Seeking latency (second)** - the time period starting from the moment a peer starts a seeking operation to the moment it starts playing the video from the seeking position after buffering 5 seconds video data.

6. **Control Overhead (Byte)** – all control message overhead including the messages of joining, seeking, data scheduling and content publishing.

We have evaluated two versions of Mediacoop using NS2 simulator. The first one is Mediacoop(no-DA), which does not include delay-awareness (DA); the second is Mediacoop(DA) with “DA”. Thus, Mediacoop(no-DA) actually does not have IPFT. Instead, it employs gossip protocol to exchange the information of peers. We separate DA in simulation
experiments in order to study the performance of the mechanisms in two stages respectively.

In order to compare Mediacoop with popular approaches, we implement PROP [6], which is a representative DHT-based VoD system. We do not implement the replica servers in PROP, because it is out of the scope of this paper. We also simulate a typical “cache and relay” system, P2VoD [5], to compare with our approach. We choose this system because it also uses playpoint to divide peers into generations. Peers in upper generation deliver data to the lower ones, organized in a tree-like structure. We add random seeking function and peer search along parents and siblings.

In our simulation, we set the video time = 3600s with playback rate = 500Kbps, and the time of a Block is 10s. We generate a transit-stub topology including 860 routers using GT-ITM [15]. And we randomly select 100 stub nodes as the IP prefix cluster routers. GT-ITM does not have the function to separate transit nodes from stub nodes, so we develop a tool to support this. The delay between any two nodes is 10ms-60ms. We generate 8,000 peers attached to IP cluster routers following uniform distribution. Work [18] proves that the user arrival follows an exponential decreasing rule rather than Poisson distribution assumed in previous studies. Therefore, we set the exponential decreasing of the peer arrival rate as:

$$\lambda(t) = \lambda_0 e^{-\tau},$$

where $\lambda_0$ is the initial arrival rate when the video published, and $\tau$ is the attenuation parameter. Accordingly, we set average inter-arrival time=5s and average peer on-line time=1800s. The download bandwidth is 1Mbps and any peer has ability to serve two streams. The startup and seeking periods both buffer 5 seconds media data.

The NS2 simulation program was performed on our super cluster, Dawning 4000A, which has 640 nodes and 2560 AMD Operton processors connected by Myrinet 2000 with 5TB in main memory [16].
Simulation is divided into 12 groups with peers from 100 to 8,000 and the total run time is about 4 days.

5.3. Simulation Results
1. Average Number of Hops. For this metric, we do not consider the gossip stage in Mediacoop(no-DA), because the purpose of comparing hops is for structured methods. We also do not mention P2VoD due to its unstructured overlay. Fig. 5 shows the average number of routing hops as a function of the overlay size from 100 to 8000 with 12 groups. The results show that PROP takes “logN” rule as the traditional DHT-based methods. Both versions of Mediacoop performance better than PROP, because the lookup hops in our method are related to the number of blocks and clusters. We have 3600/10=360 blocks and 8000/360≈22 valid clusters (maximum value) for each block, which is far less than the number of peers.

2. Control Overhead. In order to compare the update overhead of DHT-based method, we compare our scheme with PROP. As shown in Fig. 6, because Mediacoop(no-DA) adopts gossip protocol, which brings extra control message, there are more control traffic than Mediacoop(DA) as network expanding. In PROP, the overhead is considerable because peers continuously send the publishing and deleting messages. In contrast, Mediacoop(DA) can reduce the overhead, in this case, by about 40 – 70%.

3. Server Stress. There is a media content server with 1000Mbps upload bandwidth in the system to support unsatisfied peer requests. If peer has not received required data when time out occurs, or there are no supplying peers to serve it, it will ask server to send data. The lower stress on server, the more scalable the system is. We use peak stress (Mbps) on the server as the metric of system scalability. Fig. 7 shows the peak stress against different overlay size from 100 to 8000. Mediacoop(no-DA) achieves higher server stress than Mediacoop(DA). This is because some supplying peers in Mediacoop(no-DA) cause higher delays, which leads to the urgent requests time out. Then server is asked to send the missing packets again. For PROP, with buffer moving, the published pieces are discarded. This causes the requests for these pieces unsatisfied, which results in higher server stress than Mediacoop. In P2VoD, server stress increases linearly with the size of network. This is because the system is organized in a tree-like structure, with upper-level peers providing data to lower-level ones. Therefore, the parents are so lacking that most peers directly request data from the server and this makes the server stress so high.

4. Playback Continuity. Firstly, we track the playback continuity with different overlay sizes in Fig. 8. Then we examine it against simulation time with 4000 peers, as shown in Fig. 9. Our schemes perform better and improve the playback continuity much closer to 1.0. Besides the factor explained in Server Stress, the reason is that DHT methods must wait to publish the sharing messages until the data block is completely downloaded. This waiting leads to lower sharing level of the available data resources.

5. Startup and Seeking latency. Actually, these metrics involve two parts: the lookup latency and the buffering time. The first part is explained in Average Number of Hops. The second part depends on the quality of suppliers, which is also the critical factor for Playback Continuity. Note that, Mediacoop(DA) could find close suppliers and hence fill up its buffer more quickly. Fig. 10–13 illustrate the results. For the 5-second buffering time, Mediacoop(DA) only needs about 3.5 seconds for startup time and 2 seconds for seeking latency on average.

5.4. Implementation
We have implemented a scalable P2P system, called CoolFish [1], to provide VoD services. It has been released for several months and deployed in China Science & Technology Network (CSTNet), one of the four major networks in China. From October 2008 to February 2009, there were over 12,000 connections to our system. These log data has been utilized using the aforementioned method to optimize CoolFish. Table 1 shows an example of the CCDT of our system and the user IP prefix distribution is shown in Fig. 14.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Delay (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>159.226.40.*</td>
<td>0</td>
</tr>
<tr>
<td>202.127.200.*</td>
<td>31</td>
</tr>
<tr>
<td>210.72.15.*</td>
<td>8</td>
</tr>
</tbody>
</table>

Fig. 14. User IP prefix distribution in CoolFish.
CoolFish is able to sustain users of a par scale at a maximal bit rate 850Kbps (HD Video) by only a single common server. In general (94%), the seeking latency is less than 5 seconds, and for most of time (69%), is just around 3 seconds. We have randomly selected 100 users and made a satisfaction investigation on the user experience of CoolFish. Motivated by Mean Opinion Score (MOS) [17] on voice-quality measurement, we use User Opinion Score (UOS) to evaluate video streaming quality. UOS is expressed as a single number from 1 to 5, where 1 = intolerable, 2 = annoying, 3 = acceptable, 4 = joyful, 5 = perfect. The higher UOS, the better video quality is. Fig. 15 shows the results of our investigation on CoolFish. It can be observed that more than 60% users mark 4, which means CoolFish can bring them joyful experience. The average UOS of the system is 4.3 that is in accordance with the observation. More information can be found on CoolFish website [1].

![Fig. 15. User Opinion Score on CoolFish.](image)

6. Conclusion and Future Work

While lookup service is an essential function in P2P-VoD network, its content match and quality match requirements need to be satisfied effectively. Our Mediacoop, represented by its hierarchical lookup, provides both content and quality match satisfaction. It utilizes the efficiency of structured method but avoids huge update overhead. In addition, it exploits the physical delay and constructs a tree-like sub-overlay to find high quality suppliers. Hence, our approach can achieve very low startup and seeking latency with high continuity. Both the theoretical and simulation results confirm the effectiveness of our method. The real-world system has been performed over the Internet. Next step we would evaluate some other important performance. The experience of our service over the Internet is also very interesting to the further research.

7. Acknowledgements

This work is supported by the 973 National Basic Research Program of China (2004CB318109) and the 863 National High Tech Research and Development Plan of China (2006AA012452) and the National Natural Science Foundation of China (No.60873245 & No.60803085).

8. References