OCals: A Novel Overlay Construction Approach for Layered Streaming

Xin Xiao, Yuanchun Shi, Baopeng Zhang and Yuan Gao
Department of Computer Science and Technology
Tsinghua University, Beijing 100084, P.R.China
Email: {xiaoxin00@mails., shiy@, zbp02@mails.}tsinghua.edu.cn, gaoy03@gmail.com

Abstract — Layered streaming in overlay networks has drawn great interests since not only can it accommodate large scales of clients but also it handles client heterogeneities. However, to our knowledge, there’s still a lack of overlay construction (i.e. neighbor selection) approach suited for layered streaming, because i) In existing works neighbors are selected only based on their network conditions. However, a neighbor with good network condition may not be able to provide sufficient layers (e.g. a neighbor in the same LAN), ii) Previous works usually select “good” neighbors for the new node, ignoring that the joining of the new node could also be utilized to improve the performance of existing nodes. In this paper, OCals—a two-stage QoS aware overlay construction approach for layered streaming is proposed. The main contribution of OCals is that i) when selecting neighbor, it considers existing nodes’ network conditions and their providing layers as a whole; ii) it guarantees the QoS for the new node as well as improves the QoS for existing nodes so that with the joining of new nodes, the performance of the overlay could be consecutively improved; iii) it’s easy to implement and low time cost. Experiments demonstrate that compared with two other approaches: SCAMP (a pure random neighbor selection method) and Narada (a QoS aware method), the throughput and average packet delay of the layered streaming on top of the overlay constructed by OCals can be remarkably improved. Besides, the time spent on joining and recovery is very short.

I. INTRODUCTION

During recent years, overlay multicast, or application-layer multicast, has emerged as a promising way for multimedia delivery for it doesn’t rely on the IP-multicast support of routers. But the nature of multicast inevitably conflicts with the heterogeneities of user networks. To cope with this, overlay multicast with layered coding, or layered overlay multicast, is proposed to provide adaptive QoS for heterogeneous users. By now, layered overlay multicast has drawn many research interests such as[9][10][11][12][13].

To achieve high performance for the whole layered multicast and adaptive QoS for each participating node, a good overlay construction (i.e. neighbor selection) approach is indispensable, because the performance of the layered multicast and the QoS of each node depend critically on the ability of individual nodes to select good (multiple) neighbors. Thus, the question becomes how can individual nodes select their neighbors in a scalable and QoS aware fashion?

Firstly, compared with overlay multicast with single layer, layered overlay multicast has its own characteristic: the subscribing layers of a node are limited by its neighbors. Thus, in single-layer overlay multicast, the purpose of the neighbor selection is to find neighbors with good network conditions (e.g. good bandwidth and/or small delays). However, in layered overlay multicast, a newly joining node should find neighbors that not only have good network condition but also can supply sufficient layers. E.g. a newly joining node i has good bandwidth with an existing node j for they are in the same LAN; however, j might just have subscribed very few layers due to its existing poor connections with other nodes. We can see that it’s not beneficial for i to select j as its neighbor although their connection condition is good. Therefore, it’s necessary to consider existing nodes’ network conditions and their providing layers as a whole in the overlay construction for layered overlay multicast.

Secondly, in the previous researches that address QoS aware neighbor selection for peer-to-peer overlay such as Narada[3] and Bullet[4], the newly joining node usually firstly measures the network conditions with existing nodes, and then selects the nodes that have the best network conditions as its neighbors. Namely, the behavior of the newly joining node is to “select the best nodes in existing overlay”. Then we’d like to ask what can the new node do for the overlay? Can the performance of the existing overlay be improved by the joining of the new node? It could be imagined that if each newly joining node selects some “poor” existing nodes as its neighbors, the performance of the whole overlay will be consecutively improved. However, it’s necessary to point out that although the random neighbor selection method enables the newly joining node to select some “poor” nodes by random, it’s not the desired method because it doesn’t consider QoS at all.

In this paper, we propose OCals—a novel Overlay Construction Approach for Layered Streaming. In OCals, the above two important issues, namely i) considering existing nodes’ network conditions and their providing layers as a whole and ii) guaranteeing the QoS for the new node as well as improving the QoS for existing nodes, are carefully studied, and a two-stage neighbor selection method is proposed to cope with these issues. Extensive experiments demonstrate its effectiveness in improving the overlay throughput and shortening the delay observed by the nodes. At the same time, the time cost of the joining/recovery process is very low. We also study the stability issue in OCals, and the analysis and experimental results show that the un-stability caused by the replacement of “poor” nodes is quite low compared with the performance...
The rest of the paper is organized as follows. In Section II, the related works are presented. The OCals approach is described in Section III. We analyze some important issues related to OCals in Section IV. The experiments and the results are presented in Section V. We conclude our work in Section VI.

II. RELATED WORKS

The approaches that address overlay construction could be classified into two categories: one is the random node selection method and the other is the QoS aware node selection method. SCAMP[1] (a Gossip-based membership management protocol) and SwapLinks[2] are the representatives of random node selection method. Recent systems DONet[5] and PRO[6] employ SCAMP to construct the overlay. The random node selection method can well address the scalability problem, while it’s not suited for constructing the overlay for layered multicast that is sensitive to QoS metrics such as bandwidth and delay.

In some tree-based QoS aware systems such as NICE[7], Zigzag[8] and [9], to select its neighbors, a newly joining node first contacts a well known rendezvous point, then successively probes existing nodes until it finds its best position in the overlay tree. While this approach allows nodes to locate good neighbor, the single parent can easily become the bottleneck. In order to improve this, some researches are proposed to study neighbor selection in mesh-based overlay (e.g. Narada[3], and Bullet[4]), in which a newly joining node usually firstly measures the network conditions with existing nodes, and then selects the nodes that have the best network conditions as its neighbors. These mesh-based methods have been deployed to construct the overlay for single-layer streaming. But to construct a high performance overlay for layered streaming, further improvements are needed, as presented in Section I. Thus, in this paper, we propose OCals aimed to construct more appropriate overlay for layered streaming.

III. THE OCALS APPROACH

In this section, the OCals approach is presented in detail.

A. Preliminary Definitions

First of all, we give some preliminary definitions for OCals.

In order to reduce the cost of neighbor management, any pair of nodes in the overlay could keep at most one (overlay) connection and the data in all layers are transmitted on this connection. Besides, we assume that for a node, say $i$, its allowed maximum connection number is $m_i$.

If node $i$ and node $j$ are neighbors (i.e. there’s an overlay connection between them), and they both subscribe layer $l$, we say node $i$ and $j$ are logical partners in layer $l$. Figure 1 illustrates the overlay topology and the logical topology of some layer.

Furthermore, to ensure that the system could work in a relatively stable state, we assume that the allowed minimum number of logical partners of node $i$ in layer $l$ is $T_{i,l} (> 1)$ so that even if some partners leave or crash, there’re still some others working. This requirement is especially important for layered streaming, because the decoding of upper layers depends on the lower layers so that if all the logical partners that supply the lower layers leave or crash, the upper layers can not be recovered too.

B. Overlay Construction Phase

In this section we present the overlay construction phase of OCals.

The key idea of OCals includes two aspects: i) as presented in Section I, to guarantee the QoS of the newly joining node, both the existing nodes’ network conditions and their providing layers should be taken into account during neighbor selection. Therefore, in OCals, the newly joining node will probe and find appropriate logical partners for each layer rather than just selecting neighbors with good network conditions (See Stage 1). ii) since the overlay construction approach should guarantee the QoS for the new node and also improve the QoS for existing nodes, in OCals, the newly joining node will select both the nodes that can provide good QoS for it and those for which it can provide good QoS as its neighbors (See Stage 2). In the following, the basic QoS metrics adopted by OCals, and the two stages of it will be presented.

Basic QoS Metrics: RTT or Bandwidth? When a node selects neighbors, the end-to-end bandwidth and the RTT (Round Trip Time) are two basic QoS metrics. Compared with estimating RTT, it takes much longer time to estimate the end-to-end bandwidth, which often requires a series of probing packets to be sent (e.g. Packet Pair[16] probing and TFRC[15] method). Since it’s desired that the joining time of the new node should be as short as possible, we utilize the RTT as the basic QoS metric during the overlay construction phase; and during the data scheduling (i.e. requesting and relaying data) phase, TFRC is employed by each node to estimate the end-to-end bandwidth, according to which the node updates its layer subscriptions from its neighbors. In this paper, the data scheduling phase is not the main part, and we mainly focus on the influence of overlay construction method to the performance of the system.

Stage 1: Probing Existing Nodes. For each layer $l$ ($l \in [1, L]$), and $L$ is the total amount of layers that the streaming server provides), the newly joining node $i$ firstly checks how
many of the logical partners in layers $1 \sim l-1$ have also subscribed layer $l$.

If more than $T_{i,l}$ could be found, node $i$ just selects them as the logical partners in layer $l$ and doesn’t need to probe any other new neighbors for layer $l$.

Otherwise, it firstly contacts the global known rendezvous point to get an existing node that has subscribed layer $l$ and is nearest (in the IP address distance) to itself. Assume that node is $N_i$. Then, it probes $N_j$ as follows: i) It sends a probing packet to $N_j$. ii) After receiving the probing packet, $N_j$ sends back a replying packet that contains the list of its logical partners in layer $l$, as well as its RTTs with these partners. iii) With the replying packet received, node $i$ can calculate the RTT between itself and node $N_j$; and at the meanwhile, it gets the list of $N_j$’s logical partners in layer $l$.

The following process is similar: nodes $i$ sends probing packets to all of its newly known nodes, and each node that receives $i$’s probing packet will take the same action as $N_i$. This process continues until node $i$ has sent out $W_i$ probing packets—the maximum allowed number of probing packets. Figure 2 illustrates the probing process.

For each probing packet, there’s a threshold time $t_i$ for node $i$ to wait for the reply. If it receives reply within $t_i$, it calculates the RTT with the replying node; otherwise, it considers that the connection condition with that node is very poor. After receiving all the $W_i$ replies, or after the waiting time for the last probing packet reaches $t_i$, node $i$ begins to select neighbors as well as logical partners in layer $l$ as follows:

**Stage 2: QoS Aware Neighbor Selection.** Suppose $j$ is one of the nodes from which $i$ receives replying packets. We assume the RTT between $j$ and $i$ is $RTT_{j,i}$, the average and minimum RTT of $j$’s existing connections is $E(RTT_{j,s})$ and $\min(RTT_{j,s})$, respectively. Note that $RTT_{j,i}$, $E(RTT_{j,s})$ and $\min(RTT_{j,s})$ can be calculated by node $i$ with the received replying packet. In our approach, the existing node $j$ could become the neighbor candidate if one of the following conditions is met: i) $RTT_{j,i} \leq E(RTT_{j,s})$ (i.e. the RTT between $j$ and $i$ is less equal to the average RTT of $j$’s existing connections), and the connection number of $j$ doesn’t reach the maximum $m_j$; ii) $RTT_{j,i} \leq \min(RTT_{j,s})$ (i.e. the RTT between $j$ and $i$ is less equal to the minimum RTT of $j$’s existing connections), no matter whether the connection number of $j$ reaches $m_j$; and iii) $RTT_{j,i}$ is relatively small among all the RTTs between node $i$ and the replying nodes. If node $j$ is selected as neighbor because of i) or ii), it could be expected that the perceived QoS by $j$ would be improved; in contrast, if iii) holds, it’s beneficial for $i$ to select $j$ as the neighbor.

In a word, the final neighbors consist of two classes of nodes, namely those for which the newly joining node can provide good QoS and those that can provide good QoS for the newly joining node. Based on the above principle, the detailed algorithm for selecting neighbors is presented in Figure 3, in which two important issues are answered: one is the proportion of each class of neighbors; and the other is what action shall be taken if $RTT_{j,i} < \min(RTT_{j,s})$ but $j$’s connection number reaches the maximum $m_j$.

**Input:**
$U, V$ : sets used to record the replying nodes;
$Sel$ : set used to record the selected neighbors;

**Scheduling:**
for each node $j$ from which $i$ receives replying packet do
  if $RTT_{j,i} \leq E(RTT_{j,s})$ then
    node $i$ records $j$ in set $U$;
  else
    node $i$ records $j$ in set $V$;
  end if
end for
Sort the elements in set $U$ in ascending order according to their RTTs with node $i$;
$Sel$ $\leftarrow$ null;
while $|Sel| < \frac{T_i}{2}$ & $|U| > 0$ & $conn\_num\_of(i) < m_i$ do
  $j \leftarrow$ departure_head($U$);
  if $conn\_num\_of(j) < m_j$ then
    call become_partner($i, j, l, Sel$);
  elseif $RTT_{j,i} \leq \min(RTT_{j,s})$ then
    call become_partner($i, j, l, Sel$);
  suppose $RTT_{j,k}$ is the largest among all the RTTs of node $j$’s existing connections, then node $k$ breaks its connection with $j$ to keep $conn\_num\_of(j) \leq m_j$ and probes a new neighbor (See Figure 4) with the recovery method presented in Section III.C.
end if
end while
$V$ $\leftarrow$ merge($U, V$);
Sort the elements in set $V$ in ascending order according to their RTTs with node $i$;
while $|Sel| < \frac{T_i}{2}$ & $|V| > 0$ & $conn\_num\_of(i) < m_i$ do
  $j \leftarrow$ departure_head($V$);
  if $conn\_num\_of(j) < m_j$ then
    call become_partner($i, j, l, Sel$);
end if
end while

![Figure 2](image-url)
That is during node \(i\) which the node updates its subscription periodically. probe the end-to-end bandwidth with its neighbors, based on scheduling phase, in which the TFRC\([15]\) method is used to easy to implement, which we think is a very important feature require end-to-end bandwidth to be estimated, the algorithm is with more and more nodes joining. Besides, since it doesn’t the performance of the overlay will be continuously improved with more and more nodes joining. Besides, since it doesn’t require end-to-end bandwidth to be estimated, the algorithm is easy to implement, which we think is a very important feature for an overlay construction method.

After the joining process, the new node enters the data scheduling phase, in which the TFRC[15] method is used to probe the end-to-end bandwidth with its neighbors, based on which the node updates its subscription periodically.

There’s still an issue that should be further considered. That is during node \(i\)’s joining process, if an existing node \(j\)’s connection number reaches the maximum \(m_j\) but the network condition between node \(i\) and \(j\) is very good, i.e. \(RTT_{j,i} \leq \text{min}(RTT_{j,s})\), node \(i\) will replace the poorest neighbor of \(j\), say \(k\), and \(k\) has to find a better node to establish new connection (See Figure 4). While this method could improve the QoS of \(j\) and \(k\), it might cause un-stability to the overlay. In Section IV, we’ll make some analysis to it and point out that the un-stability is quite low compared with the benefit it brings.

C. Node Recovery Phase

When node \(i\)'s neighbor leaves or crashes, it starts the recovery phase immediately to select a new neighbor. Assume currently node \(i\)’s subscribed max layer is \(l\), thus the new neighbor is required to have subscribed at least \(l\) layers so that \(i\) will not be served with worse QoS than its previous neighbor provides. Therefore, \(i\) firstly contacts the rendezvous point to get an existing node that has subscribed layer \(l\) and is nearest to it. Then, from that node, it starts the probing process until \(W_i\) probing packets have been sent, which is the same with the joining phase. The difference from the joining phase is that in the recovery phase node \(i\) selects a new neighbor just according to the QoS the existing nodes can provide and the node that has the minimum RTT with node \(i\) is selected as the new neighbor. We take such a strategy because when its neighbor leaves or crashes, \(i\)’s QoS requirement may not be guaranteed, thus it should find a new neighbor that can provide the best QoS for it.

IV. ANALYSIS OF OCALS

In this section, we make analysis to some issues in OCals, one is the joining/recovery cost and another is the stability of the overlay.

A. The Analysis of Joining and Recovery Cost

A good overlay construction method should not only build a high performance overlay but also enable the new node to join the overlay as soon as possible. Since in OCals the new node probes logical partners in each layer, we’d like to explore whether fast joining speed could be achieved.

The variable \(rtt\) is used to denote the round trip time between two nodes and we assume that \(T_{i,i}\), the allowed minimum number of logical partners of node \(i\) in layer \(l\), is the same for every node, so-called \(T_i\). For the newly joining node \(i\), one \(rtt\) is cost for it to get \(N_i\) from the rendezvous point, and another \(rtt\) for it to get \(N_i\)’s logical partners(\(\geq T_i\)) in layer \(l\) from \(N_i\). So on and so forth, it will get \(1 + T_i + T_i^2 + ... + T_i^{k-1}\) nodes after \(k \ast rtt\) (\(k\) is an integer). Since node \(i\) finishes the probing process after sending probing packets to \(W_i\) nodes, the following condition should be satisfied to finish the probing:

\[
1 + T_i + T_i^2 + ... + T_i^{k-1} \geq W_i \quad (1)
\]

From (1) we can get

\[
k \geq \log_{T_i}(W_i(T_i - 1) + 1) \quad (2)
\]
Thus, for layer $l$, the maximum value of the time spent selecting neighbors is:

$$k \ast rtt + t_i = \left(\log_{T_l}(W_i(T_l - 1) + 1)\right) \ast rtt + t_i \quad (3)$$

where $t_i$ is the waiting time for the last probing packet.

So for all the layers, the maximum value of the spent time is:

$$\sum_{l=1}^{L} \left(\left(\log_{T_l}(W_i(T_l - 1) + 1)\right) \ast rtt + t_i\right) \quad (4)$$

Assume $T_l = 4$, $W_i = 20$, $\bar{rtt} = 0.05s$, $t_j = 0.25s$ and $L = 4$, we have the joining time is 1.6s, which is quite short.

The analysis of the recovery time is similar to the joining phase. Suppose currently node $i$’s subscribed max layer is $l$, the maximum value of the cost time for recovery could be similarly calculated as $\left(\log_{T_l}(W_i(T_l - 1) + 1)\right) \ast rtt + t_i$.

**B. The Analysis of Stability**

As presented in Section III.B, during node $i$’s joining process, if an existing node $j$’s connection number reaches the maximum $m_j$, and $i$ and $j$ have very good network condition, i.e. $RTT_{ji} \leq \min(RTT_{js})$, node $i$ will replace $j$’s poorest neighbor, say $k$, and $k$ has to find a better node to establish new connection. See Figure 4. Although this method could improve the QoS of $j$ and $k$, however, it might cause un-stability to the overlay. So it’s necessary to analyze i) is it worth replacing $k$? ii) how often does the replacement happen?

To answer i), we assume the newly selected neighbor by $k$ is $m$, and define a benefit factor as follows:

$$f = \frac{(RTT_{ji} + RTT_{km})}{2}{RTT_{kj}} \quad (5)$$

If $f < 1$, which means the average RTT of the two newly established connections is smaller than the RTT of previous connection $(k,j)$, we think the replacement is beneficial; otherwise, it’s not valuable. Obviously $RTT_{ji} < RTT_{kj}$, and if $RTT_{km} < RTT_{kj}$, we have $f < 1$. This requirement is easy to satisfy, because if $k$ can’t find a new neighbor better than $j$, then $j$ must be its best neighbor. However, we have known the connection condition between $k$ and $j$ is very poor. Therefore, the probability that $k$ can’t find a better neighbor is very small, namely, to replace $k$ in most cases can bring benefit.

To answer ii), let’s consider the conditions that should be met if the replacement would happen: $RTT_{ji} \leq \min(RTT_{js})$ and the connection number of $j$ reaches the maximum. Intuitively, we can feel this situation can not frequently occurs, which is demonstrated by the experiments in Section V.

**V. Evaluation**

We conduct extensive simulation experiments to study the impacts of the overlay construction approaches.

The experiments configurations are as follows: during the experiments, the participating nodes number varies from 100 to 1000. The underlying link-layer topology is generated by GT-ITM[14], where the famous transit-stub model is used. Both the intra-transit and intra-stub bandwidths are set to 10Mb, and the transit-stub bandwidths are set to 5Mb. On top of the link-layer topology, overlay construction approach is used to build the overlay, where only the nodes in stub-domains participate in the layered streaming, while the transit-domains act as the routers. On average there’re 2 new nodes joining the overlay per second. The streaming server, which is located at node 0, provides layered streaming of totally 8 layers with each layer 300Kbps. The other parameters are set as follows: $m_i = 12$, $t_i = 0.2s$, $T_{i,l} = 4$, and $W_i = 20$.

In the first set of experiments, we study the influence of the overlay construction approach to the performance (throughput and delay) of the overlay. Compared with our OCals approach, there’re two other methods used: one is the Gossip style approach-SCAMP, which utilize a pure random neighbor selection way; the other is the QoS aware approach Narada, in which a node adds a connection to another if the expected gain exceeds some threshold. Figure 5 illustrates the experiment results.
In Figure 5(a), the average throughput of the nodes is illustrated. SCAMP achieves the lowest throughput since it utilizes a pure random neighbor selection. Narada is the medium. Our OCals achieves the highest throughput, because when selecting neighbor, it considers existing nodes’ network conditions and their providing layers as a whole; besides, not only does it guarantee the QoS for the new node but also it improves the QoS for existing nodes. As a result, the most appropriate neighbors could be selected in OCals. Similarly, in 5(b), the average delay observed on each node is compared for the above three approaches. For the same reason, the delay in the overlay constructed by OCals is the smallest.

A good overlay construction method should be both beneficial to the overlay performance and with low time cost for the joining node. Thus, in the second experiment, we test the joining time of the new node that utilizes our OCals approach to select neighbors. The result is illustrated in Figure 6, which shows that the joining time doesn’t increase much with the scale of the participating nodes and keeps in the range from 2.5s to 3.5s. In the experiment, we also measured the mean RTT between nodes, which varies from 0.04s to 0.08s when nodes number varies from 100 to 1000. Based on the measured mean RTT we calculate the analytical result of the joining time with Formula (4), which is also shown in Figure 6. The result shows that the measured value is smaller than the analytical value, this is because when the new node probes the logical partners for an upper layer some logical partners in lower layers might also have subscribed the upper layer so that the new node can select them directly rather than probing others. Thus, the joining time could be saved.

Figure 7 shows the result of experiment 3, in which the nodes in the overlay leave or crash with the speed 1 node/sec and the recovery time for their neighbors is measured. It’s shown that the recovery cost is very low, at the order of hundreds of milliseconds. Besides, since each node has at least $T_{i,l}$ (set to 4 in our experiments) logical partners in layer $l$, the probability that all the partners leave/crash is very small. Therefore, with the high recovery speed and the sufficient connections, the nodes can work in a stable manner.

We study the impact of the “replacement” during joining phase in the fourth set of experiments. By replacement, we mean during node $i$’s joining process, if an existing node $j$’s connection number reaches the maximum $m_j$ but $i$ and $j$ have very good network condition, i.e. $RTT_{ji} \leq \min(RTT_{j*})$, node $i$ will replace $j$’s poorest neighbor, say $k$, and $k$ has to find a better node to establish new connection (see Figure 4). As analyzed in Section IV.B, the replacement can bring better QoS for both $j$ and $k$, but it can also cause un-stability. In the set of experiments, we measured the frequency that the replacement occurs and the benefits that are achieved, in which the frequency is equal to the times the replacement occurs versus the total connection number in the overlay. E.g., suppose there’re 100 nodes, and each node reaches the maximum connection number $m_i$, thus the total connection number is $100 \times m_i/2 = 600$; and assume the times the replacement occurs are 30, thus the replacement frequency is $30/600 = 0.05$. Figure 8 shows the replacement frequency as well as the benefit factor $f$ (see Formula (5)) when participating nodes vary from 100 to 1000. From it we can see that the frequency stays in $0.04 \sim 0.06$; and the value of $f$ shows the replacement can cause the RTT of the related nodes to be shorten to $50\% \sim 80\%$ of the old value. Furthermore, in Figure 9 and 10, we compare the mean throughput and delay of the overlays with and without the replacement strategy. With the replacement strategy, the mean throughput is improved by $6\% \sim 7\%$ and the mean delay observed by the nodes is shorten by $3\% \sim 4\%$.

In a short summary, as we expected the replacement can cause the QoS of the overlay to be improved; at the meanwhile, it doesn’t occur very often. Furthermore, the replaced node can utilize the recovery method to probe a new neighbor very soon, which is shown to be $300 \sim 400$ms on average (see Figure 6). Thus, the overlay could work in a high-performance and stable state.
VI. CONCLUSION

In this paper, we propose a two-stage QoS aware overlay construction approach called OCals for layered streaming. The main contribution of this paper is: i) when selecting neighbor, it considers existing nodes’ network conditions and their providing layers as a whole; ii) it guarantees the QoS for the new node as well as improves the QoS for existing nodes so that with the joining of new nodes, the performance of the overlay could be consecutively improved; iii) it’s easy to implement with low time cost for joining and recovery phase. Compared with two other classic overlay construction method: SCAMP and Narada, OCals can significantly improve the performance of the overlay. Its effectiveness and efficiency have been demonstrated by extensive experiments.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments. This work is supported by China NSF under Grant No.60773139, and Project of Digital Education for Public Service, No. 2006BAH02A24.

REFERENCES