Coordinate Live Streaming and Storage Sharing for Social Media Content Distribution

Xu Cheng, Jiangchuan Liu, *Senior Member, IEEE*, Haiyang Wang, and Chonggang Wang, *Senior Member, IEEE*

***Abstract*—The recently emerged user-generated contents (UGC) services, social networking services (SNS), as well as the pervasive wireless mobile network services have formed social media which has drastically changed the content distribution landscape. Today such UGC applications as YouTube allow any user to be a content provider, generating enormous amount of video contents that are quickly and extensively propagated on the Internet through such SNSes as Facebook and Twitter.**

Unfortunately, the existing UGC sites are facing critical server bottlenecks and the surges created by the social networking users would make the situation even worse. To better understand the challenges and opportunities therein, we investigate users' social behavior and personal preference of online video sharing from both real-trace measurement study on a popular social networking website and a user questionnaire survey. Our data analysis reveals an interesting coexistence of live streaming and storage sharing, and that the users are generally more interested in watching their friend's videos. It further suggests that even though the traffic is significant, most users are willing to share their resources to assist others, implying user collaboration is a rational choice in this context.

In this paper, we present Coordinated Live Streaming and Storage Sharing (COOLS), a system for efficient peer-to-peer posting of user-generated videos. Through a novel ID code design that embeds nodes' locations in an overlay, COOLS leverages stable storage users and yet inherently prioritizes living streaming flows. We also present the improvement of the basic overlay design. The evaluation results show that, as compared to other state-of-the-art solutions, COOLS successfully takes advantage of the coexistence of live streaming and storage sharing, providing better scalability, robustness, and streaming quality.

***Index Terms*—Live streaming, social media, storage sharing.**

I. Introduction

THE traditional TV has been migrating to the Internet in the past decade. Thanks to the development of such online TV devices as Apple TV and Google TV, and the emergence of such Internet TV providers as Netflix and Hulu, a number of high-quality television contents have been successfully delivered over the Internet. Besides the enterprise-level TV services, the user-generated content services, e.g., YouTube and Ustream, have allowed authorized providers to publish copyrighted movies and TV shows, as well as enormous general users to publish their own videos. It is known that this new generation of Internet video service has become more social with diverse user access patterns [1], and the social relations among users and videos make it a powerful vehicle for television content distribution.

Social media is defined as a group of Internet-based applications that build on the ideological and technological foundations of Web 2.0, and that allow the creation and exchange of user-generated content [2]. It has substantially changed the way organizations, communities, and individuals communicate. It is well-known that the trend of video sharing has become more social, by the integration of online social networking services such as Facebook and Twitter. Connecting people through cascaded relations, social media spreads information much faster and more extensively than conventional web portals or newsgroup services. Together with the pervasive penetration of wireless mobile networks and advanced devices (e.g., smartphones and tablets), TV-quality video contents can now be truly generated and accessed anywhere, at any time, and by any person. It reveals that YouTube mobile gets over 600 million views a day, and traffic from mobile devices tripled in 2011 [3]. This new video generation and propagation trend, beyond conventional TV channels, has brought up numerous well-known Internet memes.

Unfortunately, the sheer and ever-increasing data volume, the broader coverage, and longer access durations of video objects also present significant challenges than other types of objects, not only to the social networking website management, but also to the network traffic engineering and to the resource provisioning of external video sites. It is known that YouTube-like sites are facing critical server bottlenecks [1], and the surges created by the social networking users would only make the situation worse. In fact, even the text-based Twitter has encountered system-wide outages during some critical events, e.g., Obama's inauguration [4] and Michael Jackson's tragic death [5]. While peer-to-peer has long been advocated as a solution for TV or movie content streaming, it remains unclear whether it is doable for the user-generated videos with independent asynchronous viewers.

To better understand the challenges and opportunities therein, we investigate the social networking users' behavior from both system traces and a questionnaire survey. In particular, we examine the characteristics of video posting in social networking services based on traces collected from Renren network, the most popular Facebook-like social networking service in China. We find that social networking users watch and share a great amount of videos, and the distribution of the number is scale-

Manuscript received September 26, 2011; revised January 20, 2012; accepted May 17, 2012. Date of publication September 17, 2012; date of current version November 22, 2012. This work was supported by a Canada NSERC Strategic Project Grant, an NSERC Discovery Grant, an NSERC DAS Grant, and an MITACS Project Grant. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Oscar Bonastre.

X. Cheng, J. Liu, and H. Wang are with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A1S6, Canada (e-mail: xuc@cs.sfu.ca; jcliu@cs.sfu.ca; hwa17@cs.sfu.ca).

C. Wang is with InterDigital Inc., King of Prussia, PA 19406 USA (e-mail: cgwang@ieee.org).

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/TMM.2012.2217735

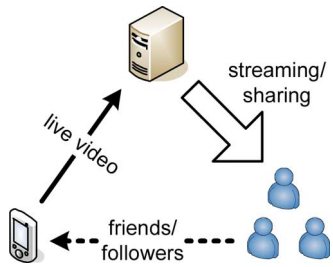


Fig. 1. Application scenario.

free. Social networking users watch a good portion of videos posted by friends, and thus together with the large amount of videos posting in the social network, this implies that client/server might suffer from lack of scalability. The measurement also shows that the interval of posting and watching video is relatively short, indicating that there is a flash-crowd after the video is posted.

Furthermore, we have also conducted a user questionnaire survey on their personal preference and social interest of Internet video sharing, to directly understand users' behavior. The survey result reveals an interesting coexistence of live streaming and storage sharing; that is, upon receiving a video post, social networking users can watch the video immediately, or download and then watch later.

As illustrated in Fig. 1, users can use the built-in camera or mobile devices to record video, and simultaneously send the live video to a server, such as YouTube and Ustream. Through the posting function of social networking services, the server can broadcast the live video to the user's friends, who can be either wired Internet users or mobile users. Upon receiving the video post, a friend has three options:

1) a friend can choose to watch the live video, and thus the requirement of streaming quality, such as startup latency and playback continuity, should be satisfied;
2) a friend can choose not to watch the live video, but she or he can download the video and expect to watch it later. Hence, such a user is considered *delay-tolerant*. In addition, the user may also switch to the first option at some time during the live streaming;
3) a friend shows no interest in the video. In this case, if such a user does not want to watch the video now or later, she or he may not want to share the resources with other uploader's friends either.

The coexistence clearly makes a system design more complicated. It however also suggests that semi-synchronized user for video streaming may reach a critical mass for collaborative streaming, and that the users downloading the video are considered relatively stable, and thus could be leveraged to combat node churns. More importantly, our survey reveals that most of the users are willing to share their resource to assist others with close relations. Also, although people are not necessarily fully satisfied with the playback quality provided by most of the current video streaming services, their concern is more about the content of the video, which largely determines the watching duration. Consequently, if their friends upload videos, they will be more interested and likely watch more of the entire video. All these features imply that collaborative peer-to-peer is a rational and promising choice in this context.

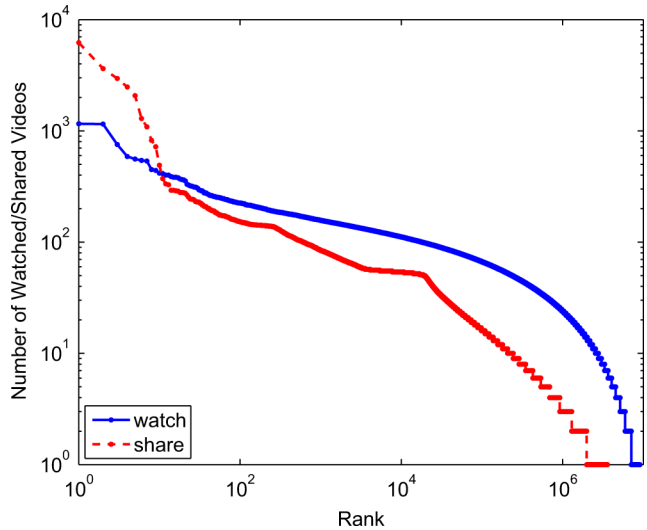


Fig. 2. Number of watched and shared videos against rank.

In this paper, we present Coordinated Live Streaming and Storage Sharing (COOLS), a system for efficient peer-to-peer posting of user-generated videos. Through a novel ID code design that embeds nodes' locations in a tree overlay, COOLS leverages stable storage users and yet inherently prioritizes living streaming flows with short startup delay. It also gracefully accommodates users' switch from storage sharing to live streaming, as well as node dynamics. We also improve our overlay tree to achieve better efficiency and robustness. The evaluation results show that, as compared to other state-of-the-art solutions, COOLS successfully takes advantage of the coexistence of live streaming and storage sharing, providing better scalability, robustness, and streaming quality.

II. Measurement Study

The social networking services have become an important media for spreading videos. Thus to understand the characteristics of video posting in social networks, we conducted a systematic measurement study on Renren network, the largest and Facebook-like social network in China. Collaborating with RenRen's engineers, we have extracted the logs from RenRen's server farm; as a result, we have collected 12.8 million records of users' posting actions and 115 million records of watching actions in one week.

When a user posts a video, her or his friends will be notified in the news feed on the social networking website. Different from text or images that can be instantly viewed, a posted video will not be really watched until the recipient clicks the link. The user can also further share the video, so that the video post will spread in the social network. We examine the number of videos a user has watched and shared. Not surprisingly, both distributions are highly skewed, displaying long-tail scale-free property, as shown in Fig. 2.

Besides the number of videos watched for each user, we also consider the number of the user's received video posts from friends. We first estimate the number of received video posts for each user, which is not readily available from the dataset. We compute the ratio of the number of watched videos and the number of received video posts from friends, defined as the

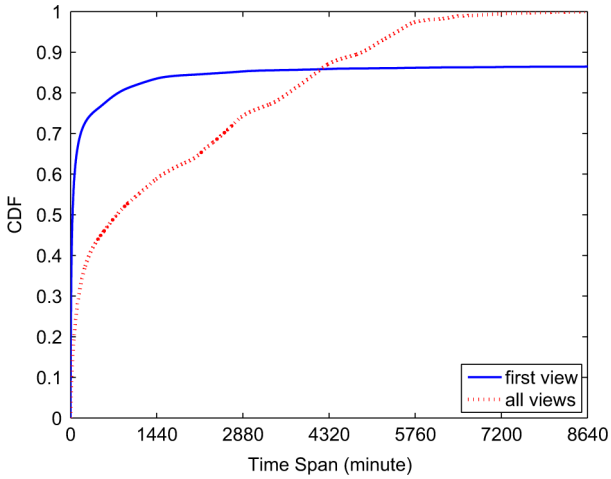


Fig. 3. CDF of time span from post to watch.

reception rate for each user. On average, users watch 16% of videos shared from friends. Although this number is not large for the individual user, considering the huge number of users and posted videos in the entire social network (refer to Fig. 2), we can observe the great number of participants for the video posting and the great number of video posts. This suggests us the client/server architecture will suffer from huge amount of usage, and also peer-to-peer delivery mechanism is a possible solution.

We also study the time span between posting a video and the actual view of this posted video by friends. We examine the sharing records that are created in the first two days, and the corresponding viewing records in within 6 days. We define the view from the first user that watches the video as *first view*, and if a shared video has not been watched in 6 days, we set the first view value as 8640 (minutes of 6 days); all the views by friends are defined as *all views*. The respective CDFs of the time spans are plotted in Fig. 3. We observe that 13% of the videos will not be watched in 6 days, and for those videos that have been watched, 68% can be watched within one hour. This indicates that videos can quickly spread to friends in the social network, exhibiting strong temporal locality.

Moreover, by examining the data of all views, we find that only 2.6% views appearing after 4 days and less than 1% after 5 days. This implies that the life span of video spreading in the social network is in general of short durations. From the figure we can also conclude that, although there are some friends accessing the video after a while, most accesses occur in a short time after the video is posted, displaying a flash-crowd property. From this conclusion, we can simplify our application scenario that we assume all the users join the system at the beginning, which is different from the conventional streaming scenario.

III. User Questionnaire Survey

We next present the user questionnaire survey results. Most of the existing studies on video sharing services measured the log traces and data crawled from the webpages to derive user-related statistics, which we have also done in Section II. Trying to further and directly understand the Internet users' preference and social interests on viewing and sharing online videos, we

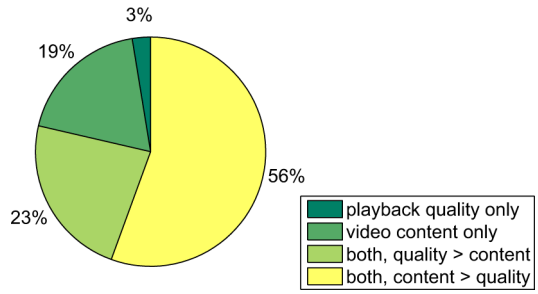


Fig. 4. Breakdown of user's concern on videos.

created a web survey and invited worldwide people to fill in. The survey contains a series of single-choice questions plus several questions on insensitive personal information. As a result, hundreds of people have participated in the survey. Of them, 59% are from North America, 33% from Asia, and 7% from Europe, with various network connections. Most of them are of ages 19–30, which is exactly the core generation of the Web 2.0 applications users.

We now summarize the key observations from the survey results. We find that 62% participants usually leave a video streaming session after selecting it and return after a while, rather than stay and wait for the startup, and 84% of them consider playback quality as the key factor of this behavior. In fact, only over half of users are satisfied with the startup latency and playback continuity, respectively. Then, users come back in a certain returning time. In terms of the time, some users consider the video length, and some consider the absolute waiting time. Considering the video length, 68% of the users come back after half of the video has been downloaded, and 22% of the users wait until the entire video is downloaded. While regardless the video length, 69% of the users spend less than 5 min for waiting, and no one will wait for more than 30 min. In short, for the same video content, *viewers of streaming and that of store-and-play both exist*.

Second, the survey asked users if they are willing to share their resources while streaming and downloading, regardless of any particular implementation (i.e., browser add-on, specific software). The result is gratified that only 11% of users do not want to contribute. Of the users, 60% do not care who they are sharing with, and 28% users only want to share the resource with close relations, e.g., friends in Facebook and mutual followers in Twitter.

Third, only 56% of the people tend to watch the entire video in general. Not surprisingly, this behavior is affected by the video content, as three quarters of users concern more about the video content than the playback quality, as shown in Fig. 4. Interestingly, when a video is uploaded by a friend, a user are more likely to watch more of the video. Fig. 5 shows the comparison of the possibility of watching the entire video, uploaded by a friend and someone the user just followed. The figure clearly implies that a video will be watched more, if it is uploaded by a user with closer relation. Unfortunately, we did not get such data on the case of watching video uploaded by a total stranger, but we believe the figure will be much lower than that of someone followed. In short, this observation, together with the fact that most of the users are willing to share their resources, indicates that user collaboration is rational in this scenario.

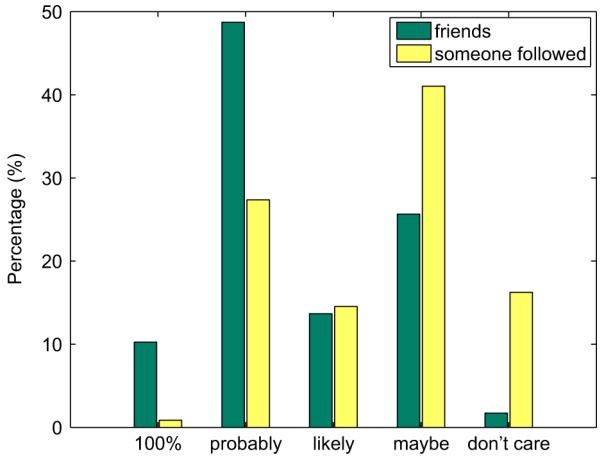


Fig. 5. Comparison of the possibility of watching the entire video.

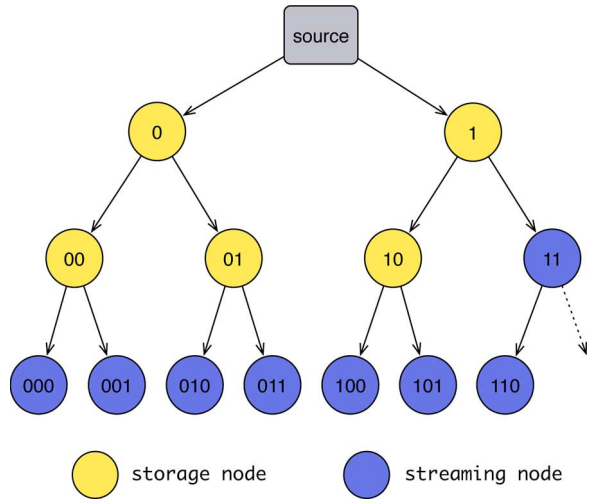


Fig. 6. Example of overlay tree with ID.

IV. COOLS System Overview

A. Streaming User and Storage User

As suggested by the survey, there exist two types of friends interested in the posted video, namely, *streaming users* and *storage users*. The streaming users expect to watch the video immediately, and the storage users expect to download and then watch the video at a different time, due to the presence of other concurrent events. When the storage users start to watch, they can either watch from the beginning or watch the current live stream, given that the live broadcast is not finished. Here, we ignore the users that are not interested in the video.

The streaming users might stop watching after a while if they find the video is out of their interest, even though the video is posted by friends. Users leaving causes dynamic and can affect the data delivery. On the other hand, the storage users that are downloading the video asynchronously do not have the concern of interest nor playback quality, until they start to watch the video, and we assume the users will not leave the system. Hence such users are considered relatively stable, though they could switch their options during downloading the video, and become streaming users in that case. Therefore, our design principle is to leverage the stable storage users to combat node churns (i.e., node dynamic behavior such as joining and leaving) in the data delivery system.

B. Overlay Tree

Considering the above factors, we advocate a tree overlay design for video posting. It is known that a tree overlay with data push is more efficient than a mesh overlay with data pull, but maintaining the tree with node churns is a daunting task. Fortunately, the existence of storage users implies that their churns are much less frequent than the traditional live streaming, which can thus be strategically placed to improve the robustness of a tree overlay.

To efficiently coordinate the two types of users, we implement a labeled tree that embeds node locations in the overlay. For ease of exposition, we explain it with a binary tree and an example is given in Fig. 6. Each node is assigned an ID, represented by a series of binary code. The two children of the root node (the source) have ID 0 and 1, respectively. For a given node, its left child's ID is the node's ID appended by a 0, and the right child's ID is that appended by a 1. As such, the ID embeds the location of a node and also that of all of its ancestors. Moreover, the number of digits (*length*) indicates its depth in the tree.

We define a partial order of the ID: if two IDs are of identical length, the one with greater value is considered greater (e.g., 010 is greater than 001); otherwise, the longer ID is greater (e.g., 000 is greater than 11). We also define an *increment* operation of the ID: if not all the bits of the ID is 1, an increment operation will increase the ID value by 1; otherwise, the length of ID will be increased by 1 and all the bits are set to 0. We also denote the value of an ID increaded by 1 as the *next value* of the ID. The operation of *decrement* can be defined similarly, while in the opposite way. We use a binary tree for easy exposition here and in the following section. The overlay tree can be extended with more children, as we will discuss in Section VI.

Since the storage nodes are relatively more stable, we expect that the storage nodes are placed at more critical locations of the tree, that is, close to the source. In other words, the storage nodes' IDs are smaller than that of streaming nodes after the tree is stabilized. Fig. 6 shows the organization of two types of nodes in the overlay tree. We will detail the construction and maintenance of the overlay in the next section, particularly on both achieving robustness with storage users and minimizing delay for streaming users.

V. COOLS Design Details

A. Overlay Construction

1) Creating Storage Tree and Streaming Tree: As mentioned, the storage nodes are expected to be close to the source. However, we also need to guarantee short startup latency for the streaming nodes, which requires them to be close to the source as well. Fortunately, since the storage users are delay-tolerant, the dilemma can be eliminated by prioritizing the streaming nodes in the initial stage.

Specifically, COOLS first constructs two trees: one contains all the streaming nodes, referred to as *streaming tree*, and the other contains all the storage nodes, referred to as *storage tree*. The source only delivers data in streaming tree at the beginning.

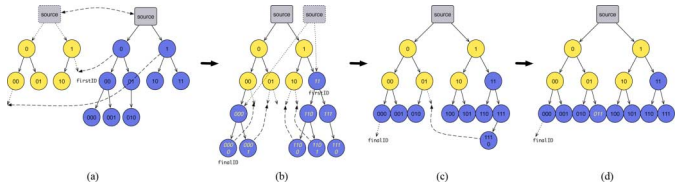


Fig. 7. Example of overlay construction: creating, merging, and promotion.

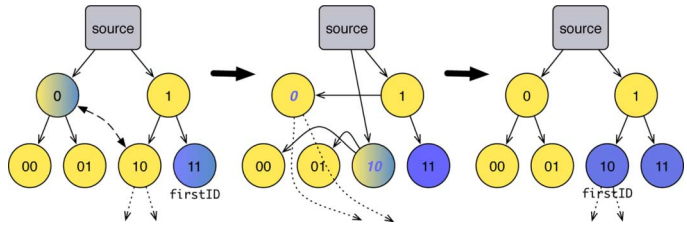


Fig. 8. Example of node demotion.

After the two trees are established, and the streaming nodes have buffered enough data to avoid timeout, the two trees will be merged to one final overlay tree.

The source records the current maximum ID of each tree. To construct the two trees, the source adds nodes to the corresponding trees sequentially. A newly added node will be assigned an ID as the next value of the maximum ID. The node thus knows its parent's location by checking the prefix of its own ID. It is worth noting that each node only keeps the local information such as the network address of the parent, two children, and the source, while the source only keeps the information of the four depth-1 children as well as the two maximum IDs. Therefore, the COOLS design shows good scalability, as the required information is independent on the number of nodes in the system.

2) Merging Tree and Node Promotion: At the beginning, the source dedicates to the streaming tree. When the streaming nodes have buffered enough data for starting playback and avoid timeout, the source starts to push video data to the storage tree. In the meantime, the source stops pushing data to the streaming tree and notifies the two depth-1 streaming tree children to connect to the parents, which are found by the source using the ID design. Since the streaming nodes have sufficient amount of the video data, they will join the storage tree seamlessly without interruption of playback. The first step in Fig. 7 shows the procedure of merging the two trees.

The source computes a potential maximum ID based on the values of the two original maximum IDs, denoted as $finalID$, e.g., 0000 in this case shown in Fig. 7. Then the source disseminates this value throughout the tree. After the two trees have been merged, the overlay tree is probably not a complete tree, as some streaming nodes may locate deeper than expected. These nodes are in an *unsteady state*, e.g., node 0000, 0001, 1100, 1101, and 1110 in the second step of Fig. 7. Some leaf storage nodes are also unsteady if they should have children but do not have yet, e.g., nodes 00, 01, and 10. Other nodes are in a *steady state*. Since most of the unsteady streaming nodes are moving upwards, we call this procedure as *node promotion*.

The unsteady nodes send control messages toward the source. Specifically, if the node finds out that its ID is no smaller than $finalID$, it will send a *promotion message*; if its potential children's ID is smaller than $finalID$ but do not have any child, the node will send a *child requiring message*. A rendezvous node (not necessary the source) receiving such messages matches them, and notifies the two senders to connect with each other. For example in Fig. 7, node 00 matches itself with node 0000, node 0 matches node 01 with node 0001, node 1 matches node 10 with nodes 1100 and 1101, and the source matches node 01 with node 1110.

B. Handling Node Dynamics

A storage user may finish her/his current event and start to watch the live video after a while, becoming a streaming node. In addition, there is also possibility that a streaming user finds the video out of interest, and thus stops watching and leaves the system. Given that the users are watching more of the entire videos that are uploaded by their friends, such events are relatively rare in our application scenario, yet proper handling of node dynamics is still necessary, as addressed below.

1) Node Demotion: For switching from storage node to streaming node, we need to demote the node in the tree. It is worth clarifying that the demotion will not degrade the playback quality; instead, it only lowers the depth of the node in the tree, since it becomes more possible to leave the overlay.

Fig. 8 shows an example of the demotion process. Supposing user 0 starts to watch the live video, it first informs the source and gets the value of $firstID$ (current maximum ID of the storage tree) from the source, and the source then decreases the value of $firstID$ by one, because one storage node is switching to streaming node. The demoting node (node 0) and the last storage node (node 10) perform a swap, so that node 0 becomes the first streaming node. Then the two nodes exchange the IDs, as well as their connections with children and parents. The demotion is then completed. Since the demoting node has already downloaded sufficient data, it can immediately start watching without any startup delay. Clearly, the demotion does not affect other nodes' playback.

2) Node Leaving and Crash: A streaming user may stop watching after finding that the video, though uploaded by a friend, is out of interest. Node crash is also possible. Storage nodes however will not leave the system unless it crashes.

When a node gracefully leaves the system, it notifies the source, parent, and children about its current information of the connections. When the source receives the notification, it will compute a new $finalID$ and disseminate the new value in the overlay. To simplify the process, all the right child nodes along the path are promoted, and all the left child nodes remain unchanged.

When a node crashes, neither the source nor its connected nodes will be notified. However, with our ID design, the crashed node's children can quickly locate their grandparent, and the grandparent thus can know the connection information of its grandchildren. The source is also notified by these children. Then these nodes repair the tree as if the crashed node leaves gracefully. It is worth noting that the crash of a storage node may cause that the new tree violates the requirement that all the storage nodes' IDs should be smaller than that of the streaming nodes, if the crashed storage node's right child is a streaming

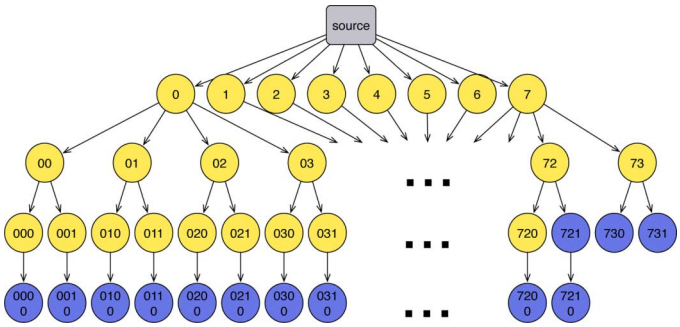


Fig. 9. Example of improved overlay tree.

node. This can be addressed by the source through computing a new $firstID$ value, and switching the last storage node and the streaming node that should be demoted.

VI. Improving COOLS Overlay Tree

In the basic COOLS overlay tree exposited above, each node has only two children. The benefit is that the node can devote more bandwidth to each children, and the overlay structure is easy to implement. However, the tree height can be too high if there are a large number of nodes. Particularly, the tree height is calculated as $\lfloor \log_k((N+1)\cdot(k-1))\rfloor$, where N is the total number of nodes in the system (considering the root is the server and does not count as a node), and k is the maximum number of children the tree node has. If there are 1000 nodes, the tree height can reach to 10, making the tree vulnerable to node dynamics.

On the other hand, even though increasing the number of children for each node can reduce the height, simply doing so will lead to another problem: the number of nodes in each depth is growing linearly, and that will burden the nodes that are close to the root. Therefore, we expect a novel tree structure, in which both the tree height and the number of nodes in each depth grow sub-linearly.

We present a novel improved overlay tree structure: if the root node has 2^k children ($k \geq 0$), then the nodes at depth i have at most 2^{k-i} children, and the tree height is no greater than $k+1$. Fig. 9 shows an example of a part of an overlay tree with height of 4. In the example, the root node has 8 children, the nodes at depth 1 have 4 children, the nodes at depth 2 have 2 children, and so on. Given that a complete tree in which the root node has 2^k children, there are 2^k nodes at depth 1, $2^k \cdot 2^{k-1}$ nodes at depth 2, and so on. Thus the number of nodes in depth i is at most $N_i = \prod_{j=1}^{i} 2^{k-j+1}$. Also the total number of nodes (excluding root) in the tree can be calculated as $\sum_{i=1}^{k} N_i$.

There is a possibility that a greater N leads to a shorter tree. To understand this, we assume a complete tree in which the root has 4 ($k=2$) children, and thus there are at most 20 nodes and the tree height is 3 ($4+4\cdot2+4\cdot2\cdot1=20$). If there are 21 nodes, the root node has to have 8 children to satisfy the requirement (k becomes 3). As a result, the new tree's height becomes 2, yet the new tree's height can be at most 4, and the complete tree can have at most $8+8\cdot4+8\cdot4\cdot2+8\cdot4\cdot2\cdot1=168$ nodes. We can see that the tree height can stay below 5 for a large number of nodes.

Since the number of nodes at depth i is $\prod_{j=1}^{i} 2^{k-j+1}$, and that at depth $(i-1)$ is then $\prod_{j=1}^{i-1} 2^{k-j+1}$, thus the growth factor at depth i is 2^{k-i+1}. As i increases, the factor decreases, and thus the number of nodes at each depth is increasing sub-linearly, which satisfies our requirement.

Accordingly, to facilitate with the improved overlay tree, the node ID is no longer base 2 number represented by 0 and 1. In particular, at depth i of an overlay tree with height k, the ID is base 2^{k-1}, which is the number of nodes at depth 1. The tree operations (e.g., construction, demotion, and leaving) can also be adapted with marginal modification.

VII. Performance Evaluation

A. Simulation Settings

We now present our evaluation for COOLS. In our evaluation, we use the following typical metrics, which together reflect the quality of service experienced by end users and the system performance.

- **Startup delay**. It is the time taken by a node between its request of joining the overlay and receiving enough data blocks to start playing back;
- **Data loss rate**. It is defined as the fraction of the data blocks that have missed their playback deadlines;
- **Control overhead**. It is size of the control messages sent by tree node.

To compare, we have also implemented Chunkyspread-like [6] and CoolStreaming-like [7] overlays. ChunkySpread is a typical tree-based multicast algorithm. Chunkyspread is unstructured, using multiple trees to balance load among nodes. It also reacts quickly to membership changes and scales well with low overhead. On the other hand, CoolStreaming is a typical mesh-based data-driven overlay network for live video streaming, in which every node periodically exchanges data availability information with a set of partners, and retrieves unavailable data from partners. The design is not only efficient but also robust and resilient, and more importantly, CoolStreaming is scalable with bounded delay.

We simulate $N = 1000$ overlay nodes, which is a typical popular video overlay size. The playback will not start until the user has obtained sufficient data (10 seconds of video data). We run the simulation 100 times for each overlay to get the average results. The survey results in Section III are applied to simulate the session setting and the node dynamics (survey results are revisited in the parentheses):

1) the session length is set to $L = 1800$ seconds and each data block is of one-second video data (no user will wait more than 30 min);
2) 600 nodes are storage nodes at the beginning (62% of users will leave and return); a storage node switches to streaming nodes with a probability p_1 at time t_i. Among the users, 30% return within 60 seconds, 40% return after 60 seconds but before 300 seconds, 20% return after 300 seconds but before 600 seconds, 10% return after 600 seconds;
3) A streaming node leaves the system with a probability of p_2 at time t_i. Among the users, 13% leave after watching $1/4$ of the video/session, 15% leave after $1/2$, 16% leave after $3/4$, and 56% watch the entire video.

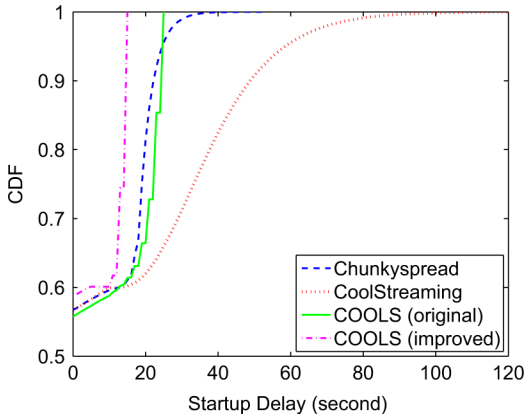


Fig. 10. CDF of startup delay.

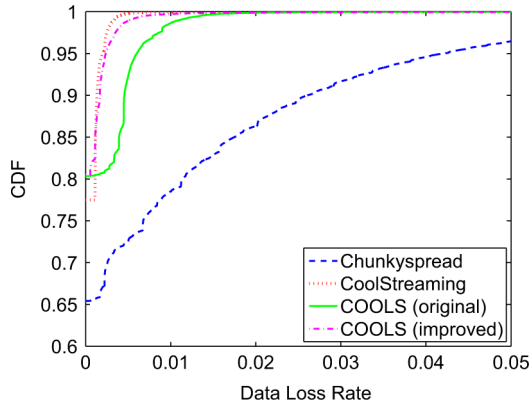


Fig. 11. CDF of data loss rate.

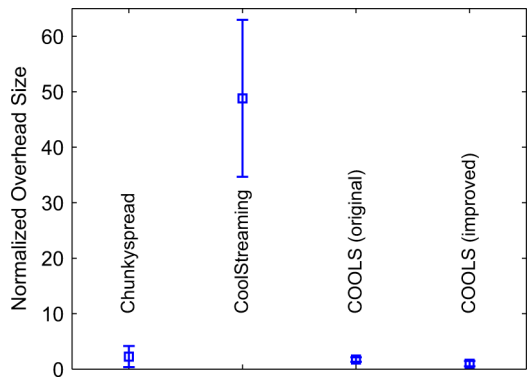


Fig. 12. Comparison of overhead size.

B. Evaluation Results

Fig. 10) shows the cumulative distribution function (CDF) of the startup delay. Because the storage nodes have already buffered some video data before switching to streaming node, there are nearly 60% nodes having no startup delay for all the solutions. For the streaming nodes from the beginning, the mesh-based solution performs worst, because it needs a longer time to search and request for partners. The pure tree-based solution and the original COOLS perform similarly, as most of the nodes need 20 seconds to startup, but a small portion of nodes need much longer time to startup in the pure tree-based solution. This is because our COOLS solution is aware of the coexistence of the two types of nodes and explicitly prioritizes the service to the streaming nodes at the beginning. Since the improved COOLS overlay is shallower, nodes need shorter time to startup than the original COOLS.

Fig. 11 shows the CDF of the data loss rate. The mesh-based solution performs the best, because it is pull-based and thus is resilient to the node dynamics. On the other hand, the pure tree-based solution performs the worst, because the tree overlay is prone to suffer from the node dynamics, as there are more than 20% nodes that have lost more than 1% data. Although our COOLS solution is also tree-based, through differentiating and leveraging the two types of nodes, specifically, placing stable storage nodes closer to the root, it performs much better than the pure tree-based solution. By further improving the overlay structure, the improved COOLS can achieve the performance of the mesh-based solution.

Finally, Fig. 12 compares the control message overhead of the four solutions. Not surprisingly, with data pull, the mesh-based solution suffers from much higher overhead than the others, as it has nearly 50 times larger overhead size. The original COOLS solution has almost the same overhead as the pure tree-based solution, while improved COOLS solution has slightly less overhead. This is because 1) by node ID design, the tree overlay is well-structured, and 2) the nodes that are close to the root are less dynamic; therefore, the nodes spend less overhead on maintaining the overlay.

VIII. Related Work

Web 2.0 applications have been emerging in the recent years, and there have been quite a few related measurement studies, particularly on understanding user-generated content services such as YouTube for video sharing [1], [8]–[10], social networking services such as Facebook and Twitter [11], [12], etc. Besides measuring to indirectly infer user behavior and social interests like these works did, we have further conducted an online questionnaire survey that directly obtains such data. The survey also motivates our study on the coexistence of live streaming and storage sharing, and implies that peer-to-peer is a rational choice in this new context. Besides peer-to-peer, client-server model has been well developed [13].

Numerous multicast and peer-to-peer protocols have been developed for live or on-demand video streaming, which can be broadly classified into two categories according to their overlay structures [14], namely, tree-based (e.g., Chunkyspread [6]) and mesh-based (e.g., CoolStreaming [7]). We have seen earlier attempts toward joint live and on-demand peer-to-peer streaming. For example, BitTorrent has enabled a streaming mode, so that user could watch on-demand video while downloading it [15]; also, peer-to-peer streaming platforms such as PPLive now provide both live and VoD modes [16]. However, efficient implementation and, more importantly, seamless integration of the two types of users, particularly in the social network context, remains a great challenge. Finally, we have extended our previous work [17] by investigating real trace from RenRen network, and improving the overlay tree with more sophisticated design that reduces the tree height.

IX. Conclusion and Future Work

This paper presents COOLS. The COOLS design is motivated by a real-trace measurement study and a questionnaire

survey on user behavior of Internet video sharing, which reveals a coexistence of live streaming and storage sharing for social media content and the interest of different users. Through a novel ID code design that inherently reflects nodes' locations in an tree overlay, COOLS leverages stable storage users and yet inherently prioritizes living streaming flows, providing better scalability, robustness, and streaming quality.

COOLS promotes the coexistence of the two types of users for video streaming. There are many possible avenues to explore in this framework. To name one, since there are wired Internet users and wireless mobile users, their heterogeneity needs to be addressed in the overlay construction and maintenance. Specifically, Internet users have better network connection, and thus they should be considered to be located close to the source. Moreover, since mobile users are usually charged for data usage, such users are not always suitable for relaying data. These requirements call for sophisticated design of the architecture as well as the operations.

REFERENCES

[1] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of YouTube videos," *Proc. IWQoS*, 2008.

[2] A. M. Kaplan and M. Haenlein, "Users of the world, unite! the challenges and opportunities of social media," *Bus. Horizons*, vol. 53, no. 1, pp. 59–68, 2010.

[3] YouTube Statistics. [Online]. Available: http://www.youtube.com/t/press_statistics.

[4] C. Albanesius, Inauguration: Twitter Reports 5 Times Normal Tweets per Second, 2009. [Online]. Available: http://appscout.pcmag.com/social-networking/273862-inauguration-twitter-repo%rts-5-times-normal-tweets-per-second.

[5] J. Tartakoff, Twitter Search Fails Under Thursday's Celebrity News Rush, 2009. [Online]. Available: http://paidcontent.org/article/419-twitter-search-fails-under-thursdays-cele%brity-news-rush.

[6] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous unstructured tree-based peer-to-Peer multicast," in *Proc. IEEE ICNP*, 2006.

[7] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "Coolstreaming/DONet: A data-driven overlay network for peer-to-Peer live media streaming," in *Proc. INFOCOM*, 2005.

[8] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system," in *Proc. IMC*, 2007.

[9] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "YouTube traffic characterization: A view from the edge," in *Proc. IMC*, 2007.

[10] A. Mislove, M. Marcon, K. Gummadi, P. Dreschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proc. IMC*, 2007.

[11] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida, "Characterizing user behavior in online social networks," in *Proc. IMC*, 2009.

[12] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?," in *Proc. WWW*, 2010.

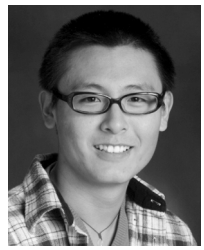
[13] NanoDataCenters Project. [Online]. Available: http://www.nanodatacenters.eu.

[14] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer Internet video broadcast," *Proc. IEEE*, vol. 96, no. 1, pp. 11–24, Jan. 2008.

[15] TorrentFreak, BitTorrent Launches Ad Supported Streaming, 2007. [Online]. Available: http://torrentfreak.com/bittorrent-launches-ad-supported-streaming-071218.

[16] Y. Huang, T. Z. Fu, D.-M. Chiu, J. C. Lui, and C. Huang, "Challenges, design and analysis of a large-scale P2P-VoD system," in *Proc. ACM SIGCOMM*, 2008, p. 2.

[17] X. Cheng and J. Liu, "Tweeting videos: Coordinate live streaming and storage sharing," in *Proc. NOSSDAV*, 2010.



Xu Cheng received the B.Sc. degree from Peking University, Beijing, China, in 2006 and the M.Sc. and Ph.D. degrees from Simon Fraser University, Burnaby, BC, Canada, in 2008 and 2012, under the supervision of Dr. Jiangchuan Liu.

His research interests include multimedia networks, social networks, and overlay networks. He is currently a research engineer in BroadbandTV Corp, Vancouver, BC, Canada.

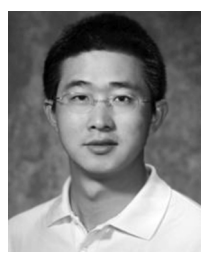
Jiangchuan Liu (S'01–M'03–SM'08) received the B.Eng. degree (cum laude) from Tsinghua University, Beijing, China, in 1999 and the Ph.D. degree from The Hong Kong University of Science and Technology in 2003, both in computer science.

He is currently an Associate Professor in the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, and was an Assistant Professor in the Department of Computer Science and Engineering at The Chinese University of Hong Kong from 2003 to 2004. His research interests include multimedia systems and networks, wireless ad hoc and sensor networks, and peer-to-peer and overlay networks.

Dr. Liu is a member of Sigma Xi. He is an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA and an editor of *IEEE Communications Surveys and Tutorials*. He is TPC Vice Chair for Information Systems of IEEE INFOCOM'2011. He is a recipient of Microsoft Research Fellowship (2000), Hong Kong Young Scientist Award (2003), and Canada NSERC DAS Award (2009). He is a co-recipient of the Best Student Paper Award of IWQoS'2008, the Best Paper Award (2009) of IEEE ComSoc Multimedia Communications Technical Committee, and Canada BCNet Broadband Challenge Winner Award 2009.

Haiyang Wang is currently pursuing the Ph.D degree in the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada.

He is working in the Multimedia and Wireless Networking Group and his research interests include cloud computing, peer-to-peer networks, multimedia systems/networks, IP routing, and QoS.



Chonggang Wang (SM'09) received the Ph.D. degree in computer science department from Beijing University of Posts and Telecommunications, Beijing, China, in 2002.

He is currently with InterDigital Communications, King of Prussia, PA, with focuses on Machine-to-Machine (M2M) communications and Internet of Things (IoT) R&D activities including technology development, standardization, and external collaboration.

Dr. Wang is the director of the IEEE ComSoc Multimedia Technical Committee E-Letter Board. He is also on the editorial board for *IEEE Communications Magazine*, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, *ACM/Springer Wireless Networks Journal*, and *Wiley Security and Communication Networks Journal*. He served and is serving as organization committee member and/or TPC member for many IEEE conferences.