CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming

Xinyan Zhang*, Jiangchuan Liu†, Bo Li‡, and Tak-Shing Peter Yum*

*Department of Information Engineering
The Chinese University of Hong Kong,Shatin, N.T., Hong Kong

†School of Computing Science
Simon Fraser University,Vancouver, BC, Canada

‡Department of Computer Science
Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong

***Abstract*— This paper presents DONet, a Data-driven Overlay Network for live media streaming. The core operations in DONet are very simple: every node periodically exchanges data availability information with a set of partners, and retrieves unavailable data from one or more partners, or supplies available data to partners. We emphasize three salient features of this data-driven design: 1) *easy to implement*, as it does not have to construct and maintain a complex global structure; 2) *efficient*, as data forwarding is dynamically determined according to data availability while not restricted by specific directions; and 3) *robust and resilient*, as the partnerships enable adaptive and quick switching among multi-suppliers. We show through analysis that DONet is scalable with bounded delay. We also address a set of practical challenges for realizing DONet, and propose an efficient member- and partnership management algorithm, together with an intelligent scheduling algorithm that achieves real-time and continuous distribution of streaming contents.**

We have extensively evaluated the performance of DONet over the PlanetLab. Our experiments, involving almost all the active PlanetLab nodes, demonstrate that DONet achieves quite good streaming quality even under formidable network conditions. Moreover, its control overhead and transmission delay are both kept at low levels.

An Internet-based DONet implementation, called *CoolStreaming v.0.9*, was released on May 30, 2004, which has attracted over 30000 distinct users with more than 4000 simultaneously being online at some peak times. We discuss the key issues toward designing *CoolStreaming* in this paper, and present several interesting observations from these large-scale tests; in particular, the larger the overlay size, the better the streaming quality it can deliver.

I. Introduction

With the widespread penetration of broadband accesses, multimedia services are getting increasingly popular among users and have contributed to a significant amount of today's Internet traffic. Many multimedia applications, such as NetTV and news broadcast, involve live media streaming from a source to a large population of users [18]. For these applications, IP Multicast is probably the most efficient vehicle; its deployment however remains confined due to many practical and political issues, such as the lack of incentives to install multicast-capable routers and to carry multicast traffic. Researchers thus have resorted to application-level solutions, which build an overlay network out of unicast tunnels across cooperative participating users, called *overlay nodes*, and multicast is then achieved through data relaying among these nodes.

Initially as remedies to IP multicast, many overlay construction algorithms also advocate a tree structure for data delivering. While this works well with dedicated infrastructure routers as in IP multicast, it poorly matches an application-level overlay with dynamic nodes. As the autonomous overlay nodes can easily crash or leave at will, a tree is highly vulnerable, which is further aggravated with streaming applications that have high bandwidth and stringent continuity demands. Sophisticated structures like mesh and forest can partially solve the problem, but they are much more complex and often less scalable.

On the other hand, migrating the multicast functionalities to application-layer also leads to greater flexibilities; in particular, all the nodes have strong buffering capabilities and can adaptively and intelligently determine the data forwarding directions. We thus envision a *data-centric* design of a streaming overlay, where a node always forward data to others that are expecting the data, with no prescribed roles like father/child, internal/external, and upstreaming/downstreaming, etc. In other words, it is the availability of data that guides the flow directions, while not a specific overlay structure that restricts the flow directions. We believe that this data-centric design is more suitable for overlay with high dynamic nodes, particularly considering that a semi-static structure, no matter how efficient, is constantly rendered to suboptimal due to node dynamics.

To this end, we design DONet, a Data-driven Overlay

0-7803-8968-9/05/$20.00 (c)2005 IEEE

Network. The core operations in DONet are very simple: every node periodically exchanges data availability information with a set of partners, and retrieves unavailable data from one or more partners, or supplies available data to partners. We emphasize three salient features of this data-driven design: 1) *easy to implement*, as it does not have to construct and maintain a complex global structure; 2) *efficient*, as data forwarding is dynamically determined according to data availability while not restricted by specific directions; and 3) *robust and resilient*, as the partnerships as well as the periodically updated data availability information enable adaptive and quick switching among multi-suppliers. Moreover, our analytical result reveals a logarithmic relation between the overlay radius and its size, implying that DONet can scale to large networks with limited delay.

To realize the data-driven overlay for live media streaming, a set of practical challenges have to be addressed. In this paper, we discuss the key design issues of DONet, including how the partnerships are formed; how the data availability information are encoded and exchanged; and how the video data are supplied and retrieved among partners. We propose a scalable membership and partnership management algorithm together with an intelligent scheduling algorithm, which enable efficient and continuous streaming of medium- to high-bandwidth contents with low control overhead. They also evenly distribute the forwarding load among the participating nodes, and accommodate nodes with heterogeneous capabilities.

We have built a prototype of DONet, and have extensively evaluated its performance over the PlanetLab testbed [30]. Our experiments involve almost all the active PlanetLab nodes [1] across 5 continents. The results demonstrate that DONet achieves high streaming quality in terms of streaming rate and playback continuity. Meanwhile, its transmission delay and control overhead are both kept at low levels. To our knowledge, Planet-based experiments on a par scale have seldom been reported in the literature. We thus list the typical issues we have encountered in the experiments, and discuss their implications to the experimental results and, possibly, to the future development of the PlanetLab.

Finally, a public Internet-based DONet implementation, called *CoolStreaming*[2], was released on May 30, 2004, which has been used to lively broadcast sports programs offered by a free video server. While initially attracted only 20 users, till the submission of this paper, over 30000 distinct users (in terms of unique IP addresses) have tested this streaming system, and more than 4000 users have been simultaneously online at some peak times. The preliminary statistical results as well as the feedbacks from the users are quite encouraging, which also reveal two interesting facts: first, the current Internet has enough available bandwidth to support TV-quality streaming (450 Kbps); and second, the larger the data-driven overlay is, the better the streaming quality it can deliver. Both reaffirm that the proposed data-driven overlay network is a promising practical solution to multicast video distribution.

[1]PlanetLab currently has over 400 participating nodes. Since not all the nodes are obligated to be online, the number of the nodes available for our experiments, or *active nodes*, typically ranges from 200 to 300.

[2]We decide to use two different names: DONet and CoolStreaming, because the former is technically sound and the latter is "commercially" sound – currently, a majority of CoolStreaming users are non-networking researchers, or even non-researchers. It happens that CoolStreaming also has a technical meaning: Cooperative overlay Streaming.

II. Related Work

There have been significant studies on video over IP multicast in the past decade; see a survey in [18]. Recently, numerous overlay multicast systems have been proposed, which can be broadly classified into two categories [11], [27]: *proxy-assisted* and *peer-to-peer based* . In the former, a set of servers or application-level proxies are strategically placed, and a high-quality overlay can then be constructed with the assistance of these anchor nodes [1], [2], [24], [26], [28]. Our DONet, however, belongs to the second category, which does not rely on dedicated nodes, but build an overlay out of self-organized autonomous nodes. In this section, we give a brief overview of the existing overlay streaming protocols, with a focus on those following the pure peer-to-peer paradigm.

A. Tree-based Protocols and Extensions

As mentioned previously, many overlay streaming systems employ a tree structure, stemmed from IP multicast. Constructing and maintaining an efficient distribution tree among the overlay nodes is a key issue to these systems. In CoopNet [3], the video source, as the root of the tree, collects the information of all the nodes for tree construction and maintenance. Such a centralized algorithm can be very efficient, but relies on a powerful and dedicated root node. To the contrary, distributed algorithms, such as SpreadIt [10], NICE [12], and ZIGZAG [11], perform the constructing and routing functions across a series of nodes. For a large-scale network, these algorithms adopt hierarchical clustering to achieve minimized transmission delay (in terms of tree height) as well as bounded node workload (in terms of fanout degree). Still, an internal node in a tree has a higher load and its leave or crash often causes buffer underflow in a large population of descendants. Several tree repairing algorithms have been devised to accommodate node dynamics [12], [11], [23]; yet the tree structure may still experience frequent breaks in the highly dynamic Internet environment.

There are many other solutions addressing the unbalanced load or vulnerability of the tree structure. Examples include building mesh-based tree (Narada and its extensions [14], and Bullet [20]), maintaining multiple distribution trees (SplitStream [19]), and leveraging layered coding (PALS [29]) or multiple description coding (CoopNet [3]). DONet complements them by introducing a simpler and straightforward data-driven design, which does not maintain an even more complex structure, nor relies on an advanced coding scheme, though the latter might be helpful in our system as well.

B. Gossip-based Protocols

Gossip (or epidemic) algorithms have recently became popular solutions to multicast message dissemination in peer-to-peer systems [13], [22]. In a typical gossip algorithm, a

node sends a newly generated message to a set of randomly selected nodes; these nodes do similarly in the next round, and so do other nodes until the message is spread to all. The random choice of gossip targets achieves resilience to random failures and enables decentralized operations. Similar to [16], we employ a gossiping protocol in DONet for membership management. The data delivery method in DONet is also partially motivated by the gossip concept. Nevertheless, the use of gossip for streaming is not straightforward because its random push may cause significant redundancy, which is particularly severe for high-bandwidth streaming applications. In DONet, we devise a smart partner selection algorithm and a low-overhead scheduling algorithm to intelligently pull data from multiple partners, which greatly reduces redundancy.

Several pioneering works on peer-to-peer on-demand streaming (e.g., [4], [5], [6], [7], [9], [8]) are closely related to gossip, and hence to DONet as well. In such a scenario, the video data provided by some seeding nodes are spread among nodes of asynchronous demands, and one or more nodes can collectively supply buffered data to a new demand, thus amplifying the system capacity with increasing suppliers over time. DONet targets live media streaming with semi-synchronized nodes, which calls for different solutions. Yet, we have also observed strong capacity amplification in our real Internet implementation, which indirectly supports the arguments in these studies on peer-to-peer on-demand streaming.

III. Design and Optimization of DONet

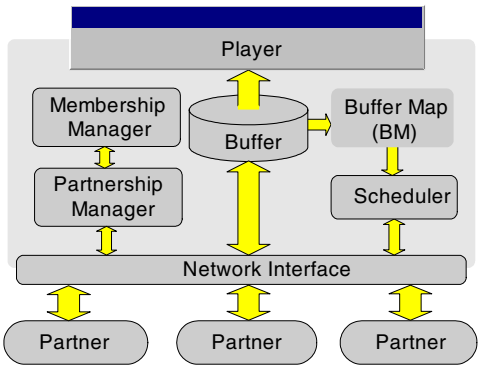

Fig. 1. A generic system diagram for a DONet node.

Figure 1 depicts the system diagram of a DONet node. There are three key modules: (1) membership manager, which helps the node maintain a partial view of other overlay nodes; (2) partnership manager, which establishes and maintains the partnership with other nodes; (3) scheduler, which schedules the transmission of video data. For each segment of a video stream, a DONet node can be either a receiver or a supplier, or both, depending dynamically on this segment's availability information, which is periodically exchanged between the node and its partners. An exception is the source node, which is always a supplier, and is referred to as the *origin node*. It could be a dedicated video server, or simply an overlay node that has a live video program to distribute.

In this section, we discuss the interactions among the modules and their design issues, and present our solutions that have been employed in the PlanetLab-based prototype as well as the real Internet-based implementation.

A. Node Join and Membership Management

Each DONet node has a unique identifier, such as its IP address, and maintains a membership cache (*mCache*) containing a partial list of the identifiers for the active nodes in the DONet. In a basic node joining algorithm, a newly joined node first contacts the origin node, which randomly selects a deputy node from its mCache and redirects the new node to the deputy. The new node can then obtain a list of partner candidates from the deputy, and contacts these candidates to establish its partners in the overlay.

This process is generally viable because the origin node persists during the lifetime of streaming and its identifier/address is universally known. The redirection enables more uniform partner selections for newly joined nodes, and greatly reduced the origin node's load. We will discuss several further enhancements to this basic algorithm in the end of this section.

A key practical issue here is how to create and update the mCache. To accommodate overlay dynamics, each node periodically generates a membership message to announce it existence; each message is a 4-tuple $<seq\_num,\ id,\ num\_partner,\ time\_to\_live>$, where *seq_num* is a sequence number of the message, *id* is the node's identifier, *num_partner* is its current number of partners, and *time_to_live* records the remaining valid time of the message. We employ the Scalable Gossip Membership protocol, SCAM, to distribute membership messages among DONet nodes. A detailed description of SCAM can be found in [21]. Here we only highlight its three desired properties: scalable, light-weight, and uniform partial view at each node. Upon receiving a message of a new *seq_num*, the DONet node updates its mCache entry for node *id*, or create the entry if not existing. The entry is a 5-tuple $<seq\_num,\ id,\ num\_partner,\ time\_to\_live,\ last\_update\_time>$, where the first four components are copied from the received membership message, and the fifth is the local time of the last update for the entry.

The following two events also trigger updates of an mCache entry: (1) the membership message is to be forwarded to other nodes through gossiping; and (2) the node serves as a deputy and the entry is to be included in the partner candidate list. In either case, *time_to_live* is decreased by *current_local_time-last_update_time*. If the new value is less than or equal to zero, the entry will be removed while not forwarded or included in the partner list; otherwise, *num_partner* will be increased by one in the deputy case.

B. Buffer Map Representation and Exchange

An example of the partnership in DONet is shown in Fig. 2 As said, neither the partnerships nor the data transmission directions are fixed in DONet. More explicitly, a video stream is divided into segments of uniform length, and the availability of the segments in the buffer of a node can be represented by

0-7803-8968-9/05/$20.00 (c)2005 IEEE

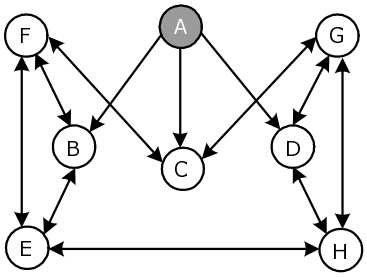

Fig. 2. Illustration of the partnership in DONet (origin node: A).

a Buffer Map (BM). Each node continuously exchange its BM with the partners, and then schedules which segment is to be fetched from which partner accordingly.

As we target live media streaming, the playback progresses of the DONet nodes are semi-synchronized. Our analytical results demonstrate that the average segment delivery latency is bounded in DONet, and the experimental results further suggest that the time lags between nodes are unlikely higher than 1 minute. Assume each segment contains 1-second video, a sliding window of 120-segment can effectively represent the buffer map of a node, because a partner is not interested in the segments that are outside of the window. As such, in our prototype, we use 120 bits to record a BM, with bit 1 indicating that a segment is available and 0 otherwise. The sequence number of the first segment in the sliding window is record by another two bytes, which can be rolled back for extra long video programs (>24 hours).

C. Scheduling Algorithm

Given the BMs of a node and that of its partners, a schedule is to be generated for fetching the expected segments from the partners. For a homogenous and static network, a simple round-robin scheduler may work well, but for a dynamic and heterogeneous network, a more intelligent scheduler is necessary. Specifically, the scheduling algorithm strikes to meet two constraints: the playback deadline for each segment, and the heterogeneous streaming bandwidth from the partners. If the first constraint cannot be satisfied, then the number of segments missing deadlines should be kept minimum. This problem is a variation of the *Parallel machine scheduling*, which is known NP-hard [25]. It is thus not easy to find an optimal solution, particularly considering that the algorithm must quickly adapt to the highly dynamic network conditions. Therefore, we resort to a simple heuristic of fast response time.

Our heuristic algorithm first calculates the number of potential suppliers for each segment (i.e., the partners containing in their buffers). Since a segment with less potential suppliers is more difficult to meet the deadline constraints, the algorithm determines the supplier of each segment starting from those with only one potential supplier, then those with two, and so forth. Among the multiple potential suppliers, the one with the highest bandwidth and enough available time is selected. A pseudo code of the scheduling algorithm at each node is shown in Fig. 3. Its complexity is bounded by $O(W \cdot B \cdot M)$, and in our implementation, each execution requires 15ms only, implying that the computation overhead is quite low. The algorithm thus can be frequently executed to update the schedule.

Input:

$band(k)$:	bandwidth from partner k;
$bm[k]$:	buffer map of partner k;
$deadlin[i]$:	deadline of segment i;
seg_size :	segment size;
$num\_partners$:	number of partners of the node;
$set\_partners$:	set of partners of the node;
$expected\_set$:	set of segments to be fetched.

Scheduling:

for segment $i \in expected\_set$ **do**
  $n \leftarrow 0$
  for j **to** $num\_partners$ **do**
    $T[j,i] \leftarrow deadline[i] - current\_time$;
      //available time for transmitting segments till i;
    $n \leftarrow n + bm[j,i]$;
      //number of potential suppliers for segment i;
  end for j;
  if $n = 1$ **then** //segments with only one potential supplier;
    $k \leftarrow \arg_r\{bm[r,i] = 1\}$;
    $supplier[i] \leftarrow k$;
    for $j \in expect\_set, j > k$ **do**
      $t[k,j] \leftarrow t[k,j] - seg\_size/band[k]$;
    end for j;
  else
    $dup\_set[n] \leftarrow dup\_set[n] \cup \{i\}$;
    $supplier[n] \leftarrow null$;
  end if;
end for i;

for $n = 2$ **to** $num\_partners$ **do**
  for each $i \in dup\_set[n]$ **do**
      //segments with n potential suppliers;
    $k \leftarrow$
      $\arg_r \Big\{ band(r) > band(r') | t[r,i] > seg\_size/band[r],$
        $t[r',i] > seg\_size/band[r'], r, r' \in set\_partners \Big\}$;
    if $k \neq null$ **then**
      $supplier[i] \leftarrow k$;
      for $j \in expected\_set, j > k$ **do**
        $t[k,j] \leftarrow t[k,j] - seg\_size/band[k]$;
      end for j;
    end if;
  end for i;
end for n;

Output:
$supplier[i]$:supplier for unavailable segment $i \in expected\_set$.

Fig. 3. Scheduling algorithm at a DONet node.

Given a schedule, the segments to be fetched from the same supplier are marked in a BM-like bit sequence, which is sent to that supplier, and these segments are then delivered in order through a real-time transport protocol. DONet does not specify a particular protocol; currently, we uses TCP protocol, as in many other systems. The BM and scheduling results can also be piggybacked by the data packets to achieve fast and low-overhead updates.

Note that the origin node severs as a supplier only, and it always has all the segments available. Provided the adaptive scheduling algorithm, it will not be overwhelmed by requests from its partners . If needed, it can also proactively control its load by advertising conservative buffer maps. For example,

assume there are M partners, the origin node can set its BM advertising to the k-th partner as

$$BM[id_{origin\_node}, i] = \begin{cases} 0, \text{if } i \bmod M \neq k \\ 1, \text{if } i \bmod M = k \end{cases}$$

that is, only the $(i \bmod M)^{\text{th}}$ partners will request segment i from the origin node, and the remaining segments will then be retrieved from other partners.

D. Failure Recovery and Partnership Refinement

In DONet, a node can depart either gracefully or accidentally due to crash. In either case, the departure can be easily detected after an idle time or BM exchange, and, as the probability of concurrent departures is rather small, an affected node can quickly react through re-scheduling using the BM information of the remaining partners. Besides this built-in recovery mechanism, we propose the following operations to further enhance resilience:

Graceful departure: the departing node should issue a departure message, which has the same format as the membership message, except that the $num\_partner$ field is set to -1.

Node failure: a partner that detects the failure will issue the departure message on behalf the failed node.

The departure message is gossiped similarly to the membership message. In the node failure case, duplicated departure messages may be generated by different partners, but only the first received will be gossiped by a node and others will be suppressed. Each node receiving the message will flush the entry for the departing node, if available, from its mCache.

Finally, we let each node periodically establish new partnerships with nodes randomly selected from its mCache. This operation serves two purposes: first, it helps each node maintain a stable number of partners in the presence of node departures; second, it helps each node explore partners of better quality. In our implementation, a node i calculates a score for its partner node j using function $\max\{\bar{s}_{i,j}, \bar{s}_{j,i}\}$, where $\bar{s}_{i,j}$ is the average number of segments that node i retrieved from node j per unit time. Intuitively, a higher outbound bandwidth and more available segments of a partner lead to a better score, and, as the partner can be either a supplier or a receiver, we shall take the maximum of both directions. After exploring new partners, the one with the lowest score can be rejected to keep a stable number of partners.

This number, M, is an important design factor. Our analytical results show that the average distance from the origin node to a destination node is bounded by $O(\log N)$, and the coverage ratio at a given distance k is $1 - e^{-\frac{M \cdot (M-1)^k - 2}{(M-2)N}}$ (see [31] for details). As an example, for a DONet of 500 nodes and $M = 4$, almost 95% of the nodes can be reached within 6 hops. More analytical and experimental results about the impact of M can be found in the following sections as well as in [31].

IV. PLANET-BASED PERFORMANCE EVALUATION

We have conducted extensive experiments with our DONet prototype. In this section, we first show the design of the experiment system in the PlanetLab environment [30], and present a set of representative results. We then identify some typical issues we have encountered and discuss their implications to the experimental results.

Fig. 4. A snapshot of the geographical node distribution.

A. Design of the Experiment System

Our experiments involve almost all the active nodes of PlanetLab, with the total number ranging from 200 to 300 during our experiment period (May to June, 2004). Each active PlanetLab node runs a copy of the prototyped program, acting as a DONet node. The origin node is located in the United States (planetlab2.lcs.mit.edu, IP: 128.31.1.12), and, through remote logins, we control the whole system at our own node in Hong Kong (planetlab2.ie.cuhk.edu.hk, IP: 137.189.97.18), which is in fact the first Asian node connected to PlanetLab (since January 2003), and is referred to as the *monitoring node*. A snap short of the geographical node distribution for an experiment in May is shown in Fig. 4.

Given the scale of this distributed testbed, effectively controlling nodes and collecting reports becomes a challenging issue, because both launching/updating the program and collecting the experimental results involve intensive login, upload, and download operations across all nodes. It is thus necessary to design an automatic control system, and the system should be highly scalable and extensible so as to easily add new nodes or new features. Interestingly, the above objectives can also be accomplished by an overlay mechanism with assistance from the tools provided PlanetLab.

We now briefly describe the major modules of the experiment system, as depicted in Fig. 5.

DONet Module: We implement the DONet module using Python, the programming language for Planet. Instead of maintaining multiple threads for concurrent events, we use an event queue with non-blocking sockets to emulate concurrent operations. Since the program is single-threaded, most difficulties related to synchronization in multi-thread programming are avoided. This makes the implementation and debug much easier, and hence enables fast prototyping.

Console and Automation Module: The console is for interactive commanding to control the whole system. Possible commands include joining DONet, leaving DONet, and tuning parameters, etc. A salient feature of Python is its support

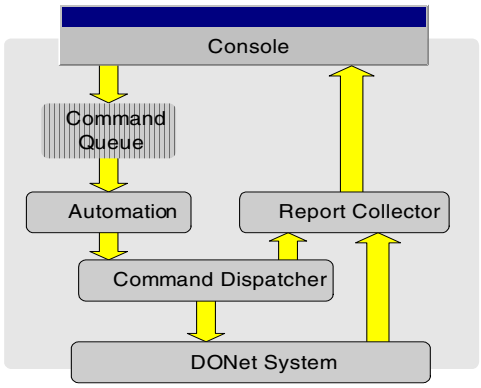

Fig. 5. A modular architecture of the experiment system.

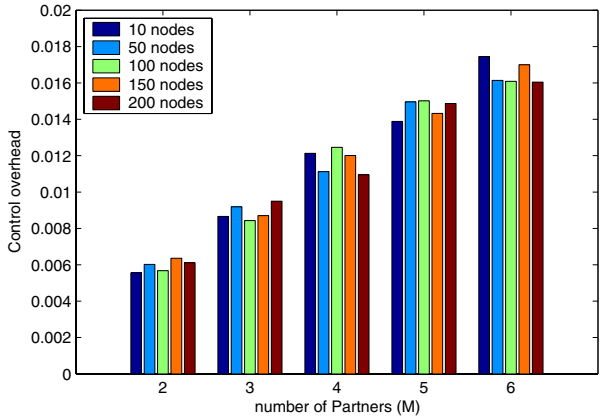

Fig. 6. Control overhead as a function of the number of partners for different overlay sizes. (Control overhead= Control traffic volume/Video traffic volume at each node).

for dynamic command executions. Thus, new features or functions can be plugged in without rewriting or reloading the whole program. We have also designed an automation in the console to automatically launch experiments and execute commands predefined in a queue. It not only achieves precisely time control, but also greatly simplifies the monitoring work for long-lasting experiments (a series of experiments often exceeds 5 hours).

Command Dispatching and Report Collecting Module: While the monitoring node can maintain a connection to each participating node to dispatch commands and collect reports, such a design is nonscalable. To mitigate this problem, we use another overlay for command dispatching. Specifically, each command message contains a unique sequence number; upon receiving a new message, a node simply sends it to a list of known nodes, which is obtained from the mCache in DONet module as well as a predefined list. As the command messages are limited and sensitive to delay, such a flooding is a reasonable choice for broadcasting commands. It also helps the construction of a reverse path tree for report collection, in which such reports as losses and path lengths can be classified and merged at some junction nodes before forwarding back to the monitoring node. Consequently, we can conduct online statistics without overwhelming the monitoring node.

Given the automatic control system, it is easy to generate stable (with persistent nodes) or dynamic environments (with dynamically joining, leaving, or failing nodes). We now present a set of representative results to demonstrate the performance of DONet under these environments and to identify the key influential factors.

B. Performance under Stable Environment

In the first set of experiments, all the nodes join in an initialization period (around 1 min) and then persist in the lifetime of the streaming (120 min, a typical length for a movie). The default streaming rate is 500 Kbps and each segment contains one second of the stream. Each DONet node maintains a sliding window of 60 segments, or 60 seconds of the streaming data, and the playback starts 10 seconds after receiving the first segment.

Control overhead: As the membership management employs a light-weight gossip protocol, most control messages in DONet are for exchanging data availability information. The number of partners thus becomes a key factor to the control overhead. Fig. 6 depicts the normalized control traffic as functions of the average number of partners. Not surprising, the overhead increases with an increase of the number of partners, but as compared to video traffic, the control traffic is essentially minor, even with over 5-6 partners (less than 2% of the total traffic). This is intuitive given that the availability of each video segment is represent by a single bit only.

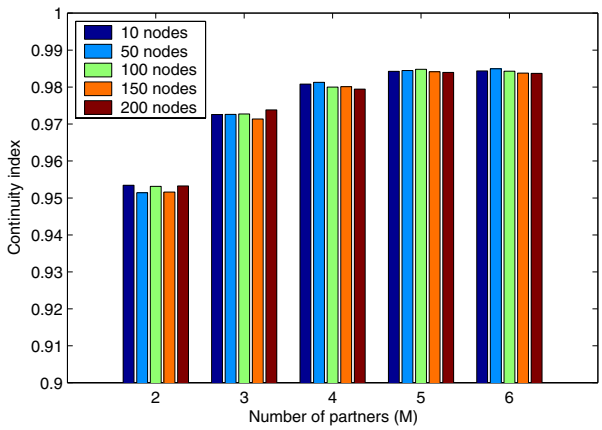

Fig. 7. Continuity index as a function of the number of partners.

Playback continuity: Maintaining continuous playback is a primary objective for streaming applications. To evaluate continuity, we define a *continuity index*, which is the number of segments that arrive before or on playback deadlines over the total number segments. Fig. 7 shows the continuity index as a function of M, the number of partners. We can see that the continuity improves with increasing M, because each node may have more choices for suppliers. The improvements with more than 4 partners are marginal. We have also shown the continuity index as function of different streaming rates in Fig.

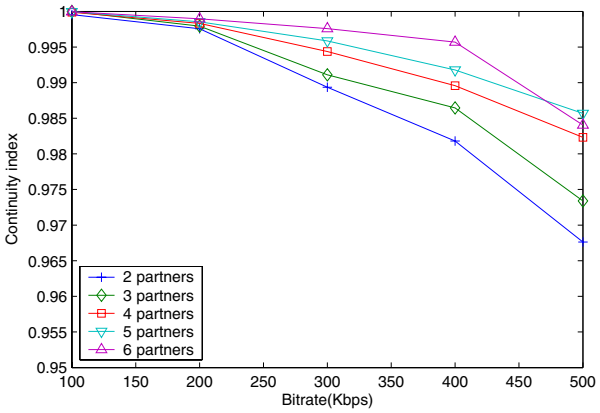

Fig. 8. Continuity index as a function of the streaming rate. Overlay size = 200 nodes.

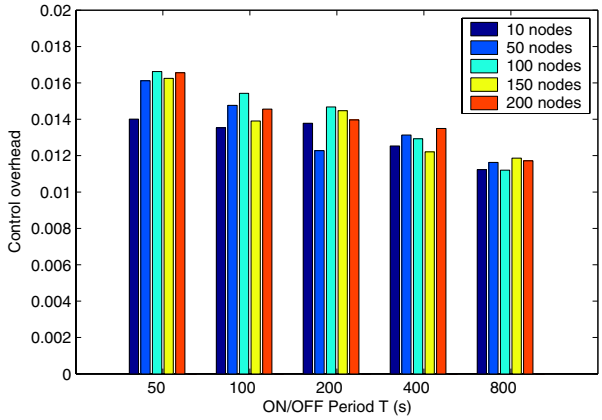

Fig. 9. Control overhead as a function of the average ON/OFF period for different overlay sizes.

8. Again, the use of 4 partners is reasonably good even under high rates. Considering that the control overhead increases with more partners, we believe that M=4 is a good practical choice, which is adopted in the following experiments.

Scalability: From Fig. 6, it can be seen that the control overhead at each node is almost independent of the overlay size. This is because the availability information (BM) are only locally exchanged. In addition, as shown in Fig. 7, the continuity index is also kept low even with large overlay sizes. In fact, we will show later that a larger overlay often leads to better playback continuity due to the increasing degree of cooperations. As summary, DONet is scalable in terms of both overlay size and streaming rate.

C. Performance under Dynamic Environment

We now examine the performance of DONet with dynamic node joining, leaving, and failing. Most parameter settings are similar to that in the previous experiments, except that each node changes its status following an ON/OFF model: the node actively participates the overlay during an ON period, and leaves (or fails) during an OFF period. Both ON and OFF periods are exponentially distributed with an average of T seconds.

Fig. 9 shows the control overhead as a function of the ON/OFF period for different overlay size. We can see that the control overhead is slightly higher with a shorter ON/OFF period (i.e., more dynamic node behaviors). Such extra control traffic is mainly contributed by the leave/failure notifications, which is only a minor part in the control traffic, as previously mentioned.

The continuity indices under different ON/OFF periods are shown in Fig. 10. Clearly, a shorter ON/OFF period leads to poorer continuity, but the drop is insignificant. With the intrinsic recovery mechanism, the continuity index of DONet remains acceptable even under highly dynamic networks (alternating in less than 1 minutes).

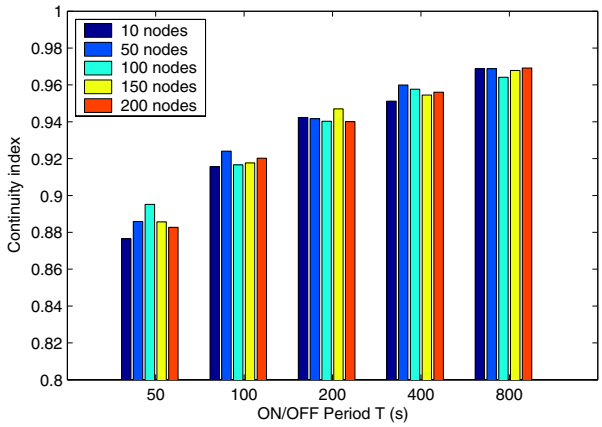

Fig. 10. Continuity index as a function of the average ON/OFF period for different overlay sizes.

D. Comparison with Tree-based Overlay

We now compared the performance of DONet with that of tree-based overlays. To achieve a fair comparison, the degree of each tree node is limited to 3; that is, except for the origin node which can have 4 children, each internal node can have a maximum of 3 children (plus one parent node, the total degree is thus 4, which is equal to the degree of a DONet node). Given the heterogeneous capability and bandwidth constraints, however, it is not always practical for a node to support 3 children; in this case, some children are moved to lower levels until the constraint is satisfied. We also employ a tree repairing method to graft downstream nodes to an upstream node when a node fails.

We first compare the end-to-end delays of DONet and the tree-based overlay. As the clocks of the PlantLab nodes are not perfectly synchronized, it is difficult to calculate the exact end-to-end delay for delivering each segment. We thus to resort an easier measure that partially reflect the delay performance, namely, overlay hop-count. The results for this measure is are presented in Fig. 12. Though it is often believed that a tree achieves shorter delay, our results show that, under

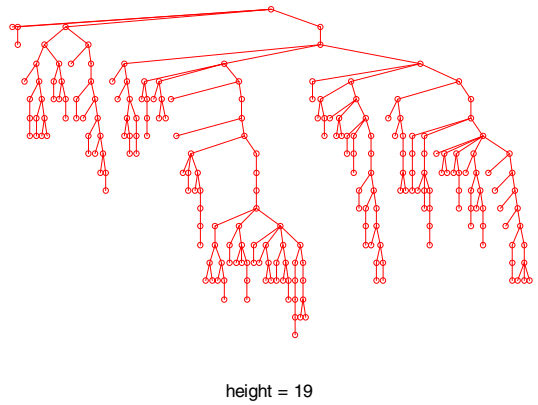

Fig. 11. A snapshot of a tree-based overlay with 231 nodes.

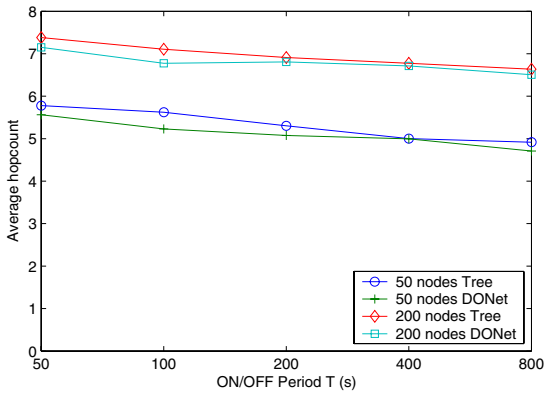

Fig. 12. Average overly hop-count of DONet and tree-based overlay.

both stable and dynamic environments, the delay measures of the tree-based overlay are slightly worse. This is because the previously motioned out-bound bandwidth constraints can noticeably increase the height of the tree. For illustration, Fig. 11 shows a snapshot of a tree in our experiment. The total number of nodes is 231, but height is 19 - recall that a full and balanced 3-ary tree of 231 nodes has a height of 5 only.

We further compare their playback continuity in Fig. 13. Clearly, the continuity index of the tree overlay is not only lower than that of the DONet, but also highly fluctuated. As an example, between 800s and 900s, there is a drop of continuity index reaching 0.4, which, according to our traces, is caused by a leave of a child of the root. As shown in Fig. 11, some internal nodes are really crucial in the tree, e.g., the rightmost child of the root as well as its own single child - either of them fails may cause buffer underflow in all the downstream nodes, which constitute more than 3/4 of the total number of nodes in the overlay. Such a problem seldom happens in DONet, as the loads of the nodes are evenly distributed and delivering paths are dynamically set according to data availability.

More comparison results as well as an analytical study can be found in [31]

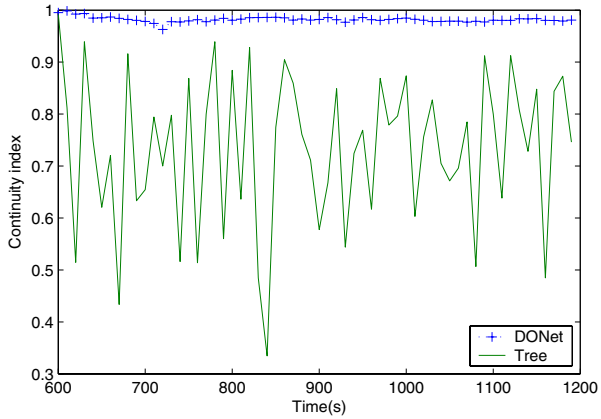

Fig. 13. Samples of continuity indices for DONet and a tree-based overlay in a experiment (from 10 min to 20 min).

E. Summary and Caveats

In summary, the performance of DONet is quite acceptable for live media streaming. Its control overhead is reasonably low, which is around 1% of the video traffic, and this ratio remains unchanged with an increase of the overlay size. As compared to a tree-based overlay, the playback continuity of DONet is much better, particularly under highly dynamic environments, and its end-to-end delay is comparable to that of the tree-based overlay.

Our experience also shows that it is easy to prototype DONet, due both to its intrinsic simplicity and the excellent support from the PlanetLab. Nevertheless, the PlanetLab is still evolving and is far from mature. We now briefly discuss some representative issues as well as their implications to our experimental results.

Scalability: As we have developed an automatic command dispatching and report collecting system, it is easy to launch the DONet program to more nodes if needed. The current scale however is limited by that of the PlanetLab. In addition, as shown in Fig. 7, most PlanetLab nodes are distributed in North America and Europe. While this partially reflects the real penetration status of the Internet, we expect a diverse environment with more nodes being deployed in other continents.

Reproducibility: Our PlanetLab-based experiments encounter the same reproducibility problem as most experimental studies that are in a *not-fully-controlled* environment. Nevertheless, the PlanetLab are reasonably stable on a time scale of several hours; the problem is thus not very severe, and we found that the results of consecutive experiments are generally comparable.

Representability: The stable network condition of the PlanetLab is partly due to less applications and hence cross traffic. In our experiments, to emulate the real Internet environment, we have intentionally added some cross traffic and throttled the injection rate in case the bandwidth is over provisioned. Another concern is about the location of the origin node. In current experiments, we selected origin nodes mainly in the

United States, for a majority of PlanetLab nodes resident there. We have also tried our own node in Hong Kong; though being remote from others, given the excellent connections between Hong Kong and North America/Europe, we have observed similar results. We are currently conducting more experiments with the origin node located in other continents as well.

V. CoolStreaming: A Practical DONet Implementation and Its Deployment Status

We have implemented a public Internet based DONet package, called *CoolStreaming*, and released the first version (v.0.9) on May 30, 2004. *CoolStreaming v.0.9* contains 2000 lines of Python source codes, and is exported from the PlanetLab-based prototype in less than 2 weeks, which reaffirms the simplicity of DONet. It currently streams Real Video and Windows Media formats, but can accommodate other streaming formats as long as they are supported by user-side players. Furthermore, its implementation is platform-independent, and thus can be used under Unix, Windows, and other operating systems.

Like other Internet applications, the success of CoolStreaming strongly relies on the content it delivers. However, unlike traditional client/server systems, an overlay system does not have a dedicated server with rich contents updated by its owners. There are other issues like copyright, and an in-depth study of them is indeed out of the scope of this research. More importantly, we have no any intention, nor ability, to become a content provider. Instead, we resort to a practical and instant solution: *DONet (CoolStreaming) as a capacity amplifier between a content provider and its clients*. In other words, it becomes part of the network infrastructure.

To this end, we have used CoolStreaming for broadcasting live sports programs ((450Kbps - 755Kbps RealVideo/Windows Media format), which are offered by a free video server. This server is happy to accept any anonymous accesses, but its capacity is very limited. Consequently, in a rush hour, most belated users cannot set up a connection directly to the server, resulting in bad experiences both to users and to the server operator. Now, by installing CoolStreaming and redirecting RealPlayer to CoolStreaming, the users can enjoy the video without connecting to and overloading the server, though with some delay, and with all the contents including the source icon and advertisements intact.[3]

While initially attracted only 20 users, till the submission of this paper, over 30000 distinct users (in terms of unique IP addresses) have installed CoolStreaming v0.9, and more than 4000 users have been simultaneously online during some peak times. If the server were directly serving these users, it would require a 3 Gbps outbound bandwidth, which is unbelievably difficult for state-of-the-art access technologies.

In Table I, we summary the distribution of the IP addresses of the CoolStreaming users. Some statistics of the number of

[3]We do not claim that our current scheme totally solves the issues of providing copyright-protected contents. For commercial content providers, authorizations are necessary and we are currently contacting with some major TV stations on such issues.

TABLE I
User IP Distribution of CoolStreaming v.0.9. (Approximation)

Time	Total	CN	HK	US	Other
June 17	1000	300	400	250	50
June 22	2400	400	1000	900	100
June 25	3000	600	1300	1000	100
June 27	4000	1000	1500	1400	100

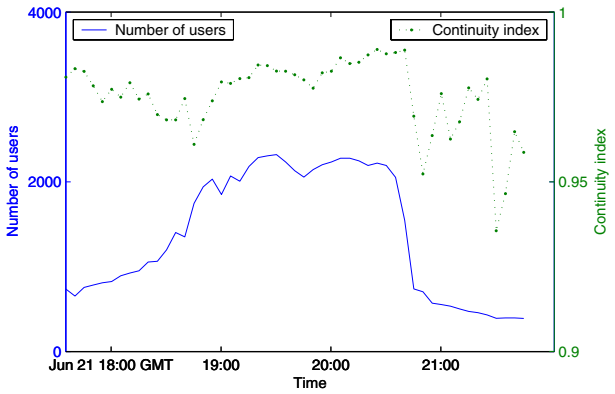

Fig. 14. Number of users and Continuity index over time.

nodes and continuity index over time in a broadcasting on July 21 are shown in Fig. 14. It can be seen that CoolStreaming achieves quite smooth playback in most of the time. More importantly, our preliminary results reveal two interesting facts:

1). The current Internet has enough available bandwidth to support TV-quality streaming ($\geq$ 450 Kbps). As a matter of fact, over 80% of the CoolStreaming users have reported (either by emails to us or through our online statistics) that the streaming is generally smooth. It confirms the speculation that the limited processing capabilities and outbound bandwidths of video servers, while not necessarily the backbone network, are slowing down the deployment of streaming services over the Internet, and overlay-based streaming, such as DONet/CoolStreaming, is a promising solution to this problem.

2. The larger the data-driven overlay is, the better the streaming quality it delivers. After releasing the first version of CoolStreaming, we did not make major updates. Interestingly, with increasing users, the statistical results as well as personal feedbacks become even better than that in the initial period. In addition, as shown in Fig. 14, the average continuity index with a higher number of nodes are generally better, though it remains over 0.95 most of the time. We conjecture that it is because the degree of cooperation increase with larger overlays, as each node has more flexibilities to locate better partners using our partner refinement algorithm. We are interest in a question on whether there is an optimal size of the data-driven overlay, and are currently examining it.

VI. Conclusions and Future Work

In this paper, we presented the design of DONet, a Data-driven Overlay Network for live media streaming. DONet does not maintain an explicit overlay structure, but adaptively forwards data according to data availability and demanding information. We discussed the key design issues of DONet, and proposed a scalable membership and partnership management algorithm together with an intelligent scheduling algorithm, which enables efficient streaming for medium- to high-bandwidth contents with low control overhead.

We extensively evaluated the performance of DONet over the PlanetLab testbed. Our experiments, involving almost all active PlanetLab nodes, demonstrated that DONet delivers quite good playback quality even under formidable network conditions. As compared with a tree-based overlay, it achieves much more continuous streaming with comparable delay.

A public Internet-based DONet implementation, called *CoolStreaming v.0.9*, was also released for broadcasting live sports programs, and has attracted over 30000 distinct users with more than 4000 simultaneously being online at some peak times. Inspired by the positive statistics and feedbacks from these users, we are currently improving our implementation and preparing for its next version. We expect to identify more potential issues and devise solutions in the future development of this project.

References

[1] S. Q. Zhuang, B. Y. Zhao, and A. D. Joseph, "Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination," in *Proc. NOSSDAV'01*, New York, Jun. 2001.

[2] L. Guo, S. Chen, S. Ren, X. Chen, and S. Jiang, "PROP: a scalable and reliable P2P assisted proxy streaming system," in *Proc. ICDCS'04*, Tokyo, Japan, Mar. 2004.

[3] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. NOSSDAV'02*, USA, May 2002.

[4] S. Jin and A. Bestavros, "Cache-and-relay streaming media delivery for asynchronous clients," in *Proc. the 4th International Workshop on Networked Group Communication (NGC)*, Boston, MA, USA, Oct. 2002.

[5] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava, "On peer-to-peer media streaming," in *Proc. ICDCS'02*, Jul. 2002.

[6] Y. Cui, B. Li, and K. Nahrstedt, "oStream: asynchronous streaming multicast," in *IEEE J. Select. Areas in Comm.*, vol. 22, Jan. 2004.

[7] Y. Cui and K. Nahrstedt, "Layered peer-to-peer streaming," in *Proc. NOSSDAV'03*, Jun. 2003.

[8] M. Heffeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: peer-to-peer media streaming using CollectCast," in *Proc. ACM Multimedia (MM'03)*, Berkeley, CA, Nov., 2003.

[9] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: peer-to-peer patching scheme for VoD service," in *Proc. WWW'03*, Budapest, Hungary, May 2003.

[10] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over peer-to-peer network," *Technical Report*, Stanford University, 2001.

[11] D. A. Tran, K. A. Hua, and T. T. Do, "A peer-to-peer architecture for media streaming," in *IEEE J. Select. Areas in Comm.*, vol. 22, Jan. 2004.

[12] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM'02*, Pittsburgh, PA, Aug. 2002.

[13] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," in *Proc. ACM SIGMETRICS'03*, San Diego, CA, USA, Jun. 2003.

[14] Y.-H. Chu, S.G. Rao, and H. Zhang, "A case for end system multicast," in *Proc. SIGMETRICS'00*, Jun. 2000.

[15] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *Proc. INFOCOM'02*, New York, USA, Jun. 2002.

[16] Y.-H. Chu, A. Ganjam, T.S. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan and H. Zhang, "Early deployment experience with an overlay based Internet broadcasting system," in *USENIX Annual Technical Conference*, Jun. 2004.

[17] X. Zhang, Q. Zhang, Z. Zhang, G. Song, W. Zhu, "A construction of locality-aware overlay network: mOverlay and its performance," *IEEE J. Select. Areas in Comm.*, vol. 22, Jan. 2004.

[18] J. Liu, B. Li, and Y.-Q. Zhang, "Adaptive video multicast over the Internet," *IEEE Multimedia*, vol. 10, no. 1, pp. 22-31, Jan./Feb. 2003.

[19] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: high-bandwidth multicast in cooperative environments," in *Proc. ACM SOSP'03*, New York, USA, Oct. 2003..

[20] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," in *Proc. ACM SOSP'03*, New York, USA, Oct. 2003.

[21] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulie, "Peer-to-peer membership management for gossip-based protocols," *IEEE Transactions on Computers*, 52(2), Feb. 2003.

[22] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie, "From epidemics to distributed computing," *IEEE Computer*, 2004 (To appear).

[23] M. Yang and Z. Fei, "A proactive approach to reconstructing overlay multicast trees," in *Proc. INFOCOM'04*, Hong Kong, Mar. 2004.

[24] M. Hefeeda, B. Bhargava, and D. K.-Y. Yau, "A hybrid architecture for cost-effective on-demand media streaming," *Computer Networks*, 44(3), Feb. 2004.

[25] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, Second Edition, MIT Press, Cambridge, MA, 2001.

[26] S. Shi and J. Turner, "Routing in overlay multicast networks," in *Proc. INFOCOM'02*, New York, Jun. 2002.

[27] S. Banerjee and B. Bhattacharjee, "A comparative study of application layer multicast protocols," under submission, available at http://www.cs.wisc.edu/ suman/pubs/compare.ps.gz

[28] Y. Chawathe, S. McCanne, and E. A. Brewer, "An architecture for Internet content distribution as an infiastmcture service," http://yatin.chawathe.com/-yafin/papers/scattercast.ps.

[29] R. Rejaie and A. Ortega, "PALS: peer to peer adaptive layered streaming," in *Proc. NOSSDAV'03*, Jun. 2003.

[30] PlanetLab Website: http://www.planet-lab.org/

[31] X. Zhang, J. Liu, B. Li and T.-S. P. Yum, "DONet/CoolStreaming: A data-driven overlay network for live media streaming," *Technical Report*, Jun. 2004.