PEER-TO-PEER ASYNCHRONOUS VIDEO STREAMING USING SKIP LIST

*Dan Wang and Jiangchuan Liu**

School of Computing Science, Simon Fraser University, Burnaby, BC,
V5A 1S6, Canada, Email: {danw, jcliu}@cs.sfu.ca

ABSTRACT

Media distribution through application-layer overlay networks has received considerable attention recently, owing to its flexibility and readily deployable nature. On-demand streaming with asynchronous requests, and in general, with VCR-like interactions, nevertheless remains a challenging task. In this paper, we introduce the Skip List, a novel randomized and distributed structure that inherently accommodates dynamic and asynchronous clients. We demonstrate a practical skip list based streaming overlay with typical VCR operations. Our simulation results show that the skip list based overlay is highly scalable, with smooth playback for diverse interactivities, and low overheads.

1. INTRODUCTION

The widespread penetration of broadband access has made the Internet of today a popular vehicle for real-time media distribution. Recently, overlay streaming [3][11] has emerged as a promising scheme for scalable streaming to a large client population. In a peer-to-peer overlay network, each node receives media data from certain neighboring nodes; the data are cached in its local buffer, and then relayed to other neighbors, eventually realizing a multicast distribution. All these operations are implemented in the application layer, which means the system is highly flexible and readily deployable.

A key challenge in an overlay streaming system is to construct a data distribution structure among the collaborative nodes. This structure should be capable of accommodating the autonomous nodes that join or leave the overlay at will or even crash without notification. The problem is further complicated when introducing on-demand playback requests, and, more general, such VCR interactions as *pause/resume*, *random-seek*, *fast-forward*, and *rewind*. These services, though attractive to clients and content providers alike, call for an additional indexing structure to locate the expected data segments with asynchronous playback offsets. This is difficult to achieve through a centralized entity, because VCR operations are more frequently invoked than node joining or leaving, and often persist a long duration. Thus, the VCR operations have seldom been incorporated in existing systems.

* J. Liu's work was supported in part by a Canadian NSERC Discovery Grant 288325, an NSERC Research Tools and Instruments Grant, a Canada Foundation for Innovation (CFI) New Opportunities Grant, a BCKDF Matching Grant, and an SFU President's Research Grant.

Several pioneering works on peer-to-peer on-demand streaming use a centralized server [4] to accommodate all asynchronous requests. This server, however, becomes a bottleneck when frequent VCR operations are introduced. In [12] the nodes are organized into a linear/tree structure with the playback offset being an indexing key. A linear or tree structure, however, can not achieve asynchronous accesses in sublinear time and require complicated rebalancing operations. Moreover, neither of them supports fast-forward and rewind.

To this end, we propose the Skip List, a novel data structure that effectively realizes the above demands. Basic skip list is first introduced in [7] and its extensions have recently been applied to practical network applications [2][6]. Specifically, Harvey *et al.* [6] have shown a scalable peer-to-peer network using SkipNet. Their focus is mainly on file indexing service as well as content and path locality. The objective and hence solutions are different from the on-demand streaming applications targeted in our study.

We demonstrate a practical overlay network that seamlessly integrates indexing and data distribution through a skip list, and discuss the key issues involved in realizing this skip list based overlay. Our simulation shows that the skip list based streaming overlay achieves a reasonably stable streaming rate with a low control overhead. More importantly, it effectively supports diverse VCR operations at a reasonably low costs, which is difficult to achieve in existing systems.

2. OVERVIEW OF SKIP LIST

A skip list [7] is a randomized ordered list of *keys* with additional, parallel links. Each key is first inserted into the *base layer* (layer 0), and then randomly *promotes* itself to the upper layer with probability $\frac{1}{2}$. If successful, the key will leave a logical node copy in the previous layer, and try to promote itself again in the new layer until it fails or a *MaxLayer* is met. Assuming it stops at layer l, it will then connect to all the neighbors from layer 0 through layer l. In this layered representation, a single key in the list is mapped into multiple logical nodes along the same column. Since the parallel links in higher layers *skip* geometrically more than those in lower layers, a key search is started from the highest layer, so as to quickly skip unnecessary parts, and then progressively move to lower layers until it hits a logical node having the key. An example of skip list with 16 nodes is shown in Fig. 1.

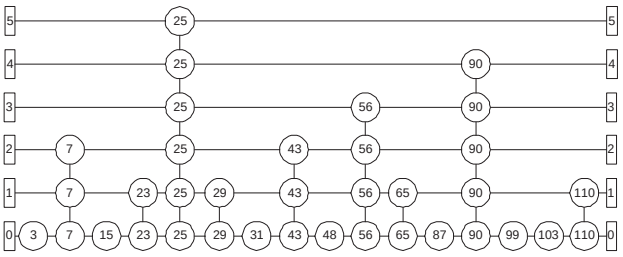

Fig. 1. A Skip List of 16 nodes.

Compared to other typical indexing structures, such as an AVL tree or a B+ tree, a randomized skip list is significantly easier to implement and generally faster. More importantly, its probabilistic nature eliminates the need for costly rebalancing operations after each key insertion, making it an attractive solution for distributed applications.

3. SKIP LIST BASED ASYNCHRONOUS OVERLAY

The playback offset of a client serves as its key in the skip list, and all logical nodes associated with this key map to the client node in the overlay. The key is updated over time according to the playback progress. Since the playing speed is identical for all the normal clients, their relative playback distances and hence the skip structure will not change over time, unless a client joins, leaves, or crashes, or a VCR operation is invoked. In addition, the client also maintains the logical links in the skip list. We detail the construction and maintenance of a skip list based overlay and the practical issues toward supporting all the typical VCR operations.

3.1. Join, Leave and Failure Recovery

We assume that there is a content server serving as an anchor node in the overlay. When a new client is to join the overlay, it first contacts the content server, which redirects the client to the top-layer node. The new client then performs a top-down search to insert itself into the base layer, and chooses the left neighbor (which has an earlier playback time) as its supplier in the overlay. It goes on to conduct the bottom-up random promotion, and set up its links to the corresponding neighbors in each layer. In this process the potential neighbors across all the layers can be recorded and a few are selected to serve as multi-partners for this new client.

Theorem 1 *The complexity of the join operation is* $O(\log N)$[1].

A client that is scheduled to leave the overlay should first notify its neighbors in the skip list, such that they can reconnect with each other to form new neighborships.

Theorem 2 *The expected message cost for a graceful client departure is* $O(1)$.

[1]The proofs of all lemmas and theorems can be found in [9].

Intuitively, this constant amortized cost holds because the number of nodes with $O(\log N)$ neighbors is quite small. Most of the nodes have far fewer neighbors in the skip list, and hence lower costs. A similar argument suggests that while the maximum extra neighborhood information kept at a client node is $O(\log N)$, the average is $O(1)$ only.

Every client periodically exchanges echo messages with its neighbors in the skip list, enabling an abrupt client failure to be easily detected. The parallel links in the skip list help the affected neighbors perform local repairs. This is a variation of the search operation and the cost is at most $O(\log N)$.

3.2. VCR-like Interactions

Since the cost for a leave and then re-join with a new playback offset is only $O(\log N)$, this combination can be used to implement most typical VCR operations.

1. Pause and Resume The client can simply stop playback, but still stay in the overlay, accepting and supplying data at the normal speed. If the pause time is very long and its buffer overflows, it temporarily leave the overlay. Once the resume command is given, it rejoins with the original offset.

2. Random-seek The client can simply leave and then rejoin the overlay with the new offset after seeking.

3. Fast-forward and Rewind Assume the speed of fast-forward or rewind is v, which can be realized by playing one segment out of v segments [10]. This can be implemented through random seek; however, since a fast-forward or rewind movement generally consists of a long series of such jump operations, more efficient solutions are expected.

Given that an overlay node has a size-limited buffer, after jumping $v-1$ segments, the fast-forwarding or rewinding client has to move from its current supplier to a new one. The key issue of a jump operation is thus to locate the suppliers with the expected segments in time. Skip list provides effective support toward this operation through its horizontal links, which enable a client to skip unnecessary nodes (and hence segments) at a fairly stable speed. We now analyze the cost of the jump operations for fast-forward. The analysis also applies to rewind movements after a symmetric transform.

We assume that the playback offsets of two consecutive nodes in the overlay differ d segments on average, and the client can retrieve n data segments from its supplier with VCR speed v. Then following lemma gives the nodes to be skipped.

Lemma 3 *The number of logical nodes to be skipped ahead is* $\frac{n(v-1)}{d}$ *for each jump operation.*

Theorem 4 *Each jump operation for fast-forward is* $O(1)$.

Intuitively, the client needs to follow layer $\log(\frac{n(v-1)}{d})$ in the skip list. The above theorem suggests that the cost for a jump operation is independent of the overlay size, nor the fast-forwarding/rewinding speed.

In practice, $\log(\frac{n(v-1)}{d})$ might not be an integer. In this case, the client can temporarily move along higher or lower layer links, or adaptively set n; i.e. stay for a longer or shorter period with the current supplier, to achieve an average speed of v. We now derive the range for n, the number of segments to be retrieved from a supplier at a fast-forwarding speed v, and the maximum speed that the overlay can support.

We consider the following physical constraints, which are adapted from [8]: B, the buffer size of each client; T, the connection time from one client to the other; t, the playing time for each data segment. We also assume that the downloading time of each segment is smaller than t.

Lemma 5 *For fast-forwarding speed v, the number of data segments that a supplier can provide is* $[\frac{T}{t}, \frac{B(B-v)}{vB-B}]$.

Theorem 6 *The maximum speed that the system can provide is no higher than* $\frac{tB^2-TB}{TB-tB}$.

Although the VCR speed of the system has a upperbound limited by the system constraints, a speed of no more than 32x can be easily achieved even in a small to medium skip list overlay (say, less than 500 nodes), and, as investigated in [10], a higher speed is rarely perceived as useful by users, nor is it supported in most commercial VHS or DVD players.

4. PERFORMANCE EVALUATION

In this section, we present a few representative results of our simulation. For a complete version, please refer to [9].

We used the GT-ITM topology generator to produce a 1000-node networks. The streaming rate was 256Kbps and the length of the stream 150 minutes. The default size of the client-side buffer was 15Mbytes. We adopted Segment Missing Rate (SMR) as the major criterion for evaluating the streaming quality. A data segment is considered missing if it is not available to client till the play-out time, and the SMR for the whole system is the average ratio of the missed segments across all the participating clients. For comparison, we also simulated an existing on-demand overlay streaming system, *oStream* [4] which employs a tree structure. Each client node caches played-out data and relays to its children, which may have asynchronous playback offsets. A centralized directory server is used to maintain the global information of the overlay, which facilitates client join or failure recovery.

4.1. Streaming Quality

Fig. 2 plots the segment loss rates (SMRs) for 1000-node skip list and *oStream* overlays during a 4500-sec simulation with cross traffic. It can be seen that the loss rate of skip list is generally less than 0.1, which is not only lower than *oStream*, but also more stable. From a video decoding point of view, such a loss can be effectively masked by interleaving or error-concealment techniques. On the other hand, the loss

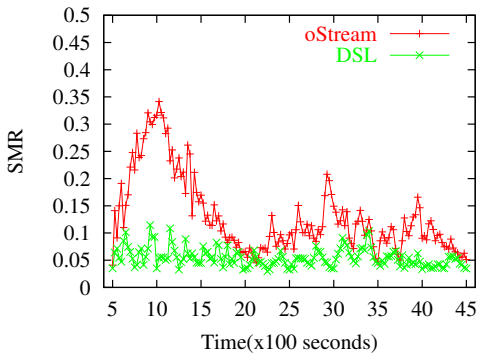

Fig. 2. Segment missing rate (SMR) for DSL (skip list) and oStream with local bandwidth fluctuation.

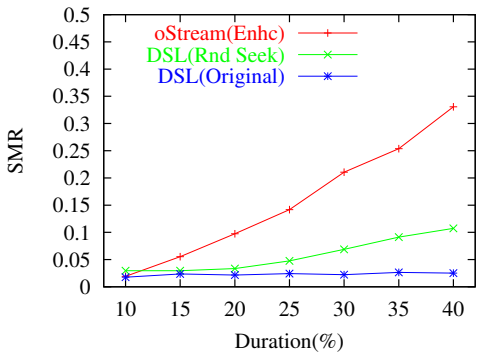

Fig. 3. Segment missing rate as a function of VCR duration for DSL (skip list) and oStream.

rate of *oStream* greatly fluctuates over time, with a peak value as high as 0.35, resulting in poor video quality. This is mainly because *oStream* relies on a specific tree structure for streaming, so bandwidth reduction at an internal link of the tree, and particularly at those close to the root, could result in severe loss across many descendants.

4.2. Impact of VCR Interactions

The current version of *oStream* does not support these operations. To enable comparison, we implemented an enhanced *oStream*, in which a jump operation is realized by moving through the links in the tree structure either in the direction of the parent (for fast-forward) or the descendant (for rewind). This distributed search avoids repeated contacts with the server, which is clearly non-scalable for frequent VCR operations.

We randomly picked up 10% of clients to perform fast-forward or rewind operations with 2x speed. The duration of each VCR operation varied from 5% of the stream length to 40%. Fig. 3 shows streaming quality for the skip list and the enhanced *oStream* overlays. Clearly, skip list outperforms *oStream*, and its quality is almost independent of the duration of the VCR operations. On the other hand, the quality of the enhanced *oStream* quickly becomes worse as duration increases, and is generally unacceptable for a duration of greater

than 20% of the stream length. Intuitively, for a tree overlay like *oStream*, the node initiating fast-forwarding or rewinding may still find the expected data segments in the buffers of its parent or close ancestors, but such an inefficient linear search will soon cause it to suffer from buffer outage.

As mentioned before, each jump can be implemented through a random-seek operation as well. To show its (in)effectiveness, we also plotted streaming quality in this implementation. As shown in Fig. 3, its performance is generally acceptable with short VCR durations, but becomes worse with increased duration. This is mainly because random-seek needs a much higher cost and hence longer time to identify subsequent suppliers, and the lag accumulates over time.

5. CONCLUSIONS AND DISCUSSIONS

In this paper, we introduced a skip list, for on-demand overlay media streaming. A skip list is a randomized and distributed structure, which inherently accommodates node dynamics and asynchronous requests. We have shown that all typical VCR operations can be implemented in this overlay with $O(1)$ or $O(\log N)$ message costs.

The performance of the skip list based overlay was examined under different network and client configurations, and our preliminary results re-affirm its excellent scalability and robustness. Its streaming quality is reasonably good with asynchronous requests and frequent VCR operations, while the latter has seldom been supported in existing overlay systems.

There are difficulties associated with the skip list, however. For example, the client which unfortunately promotes itself to the highest layer is responsible for all join dispatches. This might introduce big burdens for a client machine. The situation is made worse as the number of client is not determined in advance, and thus, the *MaxLayer* discussed in section III does not exist, resulting the top layer nodes be promoted excessively. We address these issues in [9] by introducing a novel dynamic skip list (DSL) structure which can effectively restrain the layers and balance the overhead. In DSL, we compress a suitable number of top layers of the skip list into one layer, which is monitored by the content server. This top layer is dynamic overtime. We thus design an efficient scheme for the content server to compress the top layers.

In addition to content indexing, content distribution is another important feature for peer-to-peer overlays. As we are using multi suppliers for robustness, the content distribution becomes more complicated. In [9], we design a distribution scheme based on network coding. Network coding [1] is first considered for improving network throughput. It is used [5] for content distribution of large files in peer-to-peer networks. We apply network coding in video streaming for the first time and give out an optimal content scheduling algorithm.

As a future work, we are interested in investigating the performance of the DSL-based streaming overlay with more realistic network configurations and heterogeneous clients, possibly using the PlanetLab testbed, and compare it with those overlay systems using advanced structures. We are also interested in incorporating advanced network coding or video coding algorithms; an example is the Multiple Description Coding (MDC), which is a good match to the DSL structure and could further improve its robustness.

6. REFERENCES

[1] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network Information Flow", *IEEE Transactions on Information Theory*, vol. 46, pp. 1204-1216, July, 2000.

[2] J. Aspnes and G. Shah, Skip Graphs in *Proc. ACM SODA'03*, Baltimore, MD, Jan. 2003.

[3] Y. Chu, S. Rao, and H. Zhang, "A Case for End System Multicast," in *Proc. ACM SIGMETRICS'00*, Santa Clara, CA, Jun. 2000.

[4] Y. Cui, B. Li, and K. Nahrstedt, "oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 91-106, Jan. 2004.

[5] C. Gkantsidis and P. Rodriguez, "Network Coding for Large Scale Content Distribution," in *Proc. IEEE INFOCOM'05*, Miami, FL, Mar. 2005.

[6] N. Harvey, M. Jones, S. Saroiu, M. Theimer and A. Wolman, "SkipNet: A Scalable Overlay Network with Practical Locality Properties," in *Proc. USITS'03*, Seattle, WA, Mar. 2003.

[7] W. Pugh, "Skip Lists: A Probabilistic Alternative to Balanced Trees," *Communications of the ACM*, vol. 33, no. 6, pp. 668-676, Jun. 1990.

[8] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," in *Proc. ACM SIGCOMM'04*, Portland, OR, Aug. 2004.

[9] D. Wang and J. Liu, "A Dynamic Skip List Based Overlay for Asynchronous Media Streaming," *Technical Report*, Simon Fraser University, May, 2005.

[10] B. Wildemuth, G. Marchionini, M. Yang, G. Geisler, T. Wilkens, A. Hughes, and R. Gruss, "How Fast is too Fast? Evaluating Fast Forward Surrogates for Digital Video," in *Proc. JCDL'03*, Houston, TX, May, 2003.

[11] 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. IEEE INFOCOM'05*, Miami, FL, Mar. 2005.

[12] M. Zhou and J. Liu, "Tree-Assisted Gossiping for Overlay Video Distribution," to appear in *Kluwer Multimedia Tools and Applications*, 2005.