RELookup: Providing Resilient and Efficient Lookup Service for P2P-VoD Streaming

Xu Zhang [1], Zhenhua Li [2], Tieying Zhang [3], Liangpeng He [1], and Guihai Chen [1]
[1] Nanjing University, Nanjing, China [2] Peking University, Beijing, China
[3] ICT, Chinese Academy of Sciences, Beijing, China
zxyecn@126.com, lzh@net.pku.edu.cn, gchen@nju.edu.cn

Abstract

For P2P-VoD streaming, an effective lookup algorithm for appropriate data suppliers is required to support the user's operation of random jump on the video. Existing lookup algorithms mainly adopt a centralized, flooding-based, or DHT-based method. Facing the highly dynamic Internet environments, the centralized method incurs a single point of failure, the flooding-based method lacks scalability, and the DHT-based method is not resilient. Motivated by these problems, we propose a novel lookup algorithm, named "RELookup"*, which places peers on a resilient super node-based overlay and meanwhile utilizes the* play point distance *to efficiently locate candidate data suppliers. Besides, deliberate measures (i.e., special design of message format and node state) have been taken to reduce the coordination costs between super nodes to very little. Results of trace-driven simulations confirm the effectiveness of our proposed RELookup algorithm.*

***Keywords*-P2P (peer-to-peer); VoD (video-on-demand); lookup algorithm; super node; play point distance.**

I. Introduction

In recent years, peer-to-peer video-on-demand (P2P VoD) streaming [1], [2], [3], [4], [5], [6] has attracted enormous attention from both industry and academy. As a data intensive Internet application, P2P-VoD desperately demands locating and accessing data effectively. Therefore, an effective lookup algorithm for appropriate data suppliers is required to support the user's operation of random jump on the video. The existing lookup algorithms mainly adopt a centralized [1], [2], [6], flooding-based [7], [8] or DHT-based [3], [4], [9], [10] method. Facing the highly dynamic Internet environment, the centralized method incurs a single point of failure, the flooding-based method lacks scalability because of the exponentially increasing communication cost, and the DHT-based method is not resilient due to the necessity of maintaining a structured overlay [11], [12].

Motivated by these problems, in this paper we propose a novel lookup algorithm for P2P-VoD streaming, named *"RELookup"*, which places peers on a resilient super node-based overlay and meanwhile utilizes the design of *play point distance* to efficiently locate candidate data suppliers. It is well known that organizing the peers into an unstructured (random) overlay (like Gnutella [13]) is simple, fault-tolerant, but not scalable. Therefore, following the design philosophy of tiered unstructured overlays (like eDonkey [14] and KaZaa [15]), we select a moderate number of (relatively) stable super nodes from the peers to organize the network so as to make the system resilient and scalable.

Meanwhile, in the RELookup algorithm, super nodes manage the index of data suppliers by utilizing the *(unchanged) play point distance* [10]. Each super node maintains the information of data suppliers for a range of data blocks. Peers use the *(unchanged) play point distance* to locate the candidate data suppliers with the required data blocks. Here "unchanged" means the viewers of the same video are expected to playback continuously and thus their play point distances do not change in most time. The abovementioned "unchanged distance" leads to little requirement for the state update messages between super nodes, which significantly reduces communication overhead. Additionally, deliberate measures (i.e., special design of message format and node state) have been taken to further reduce the coordination costs between super nodes to very little. In particular, every super node executes at most three flooding operations (at the time of join, split, and logout) in its working life.

The results of trace-driven simulations confirm the effectiveness of our proposed RELookup algorithm. Specifically, RELookup possesses low control overhead, low jump delay, high playback continuity, and high accuracy of node state (i.e., global information table).

II. Related Work

The existing lookup algorithms for P2P-VoD streaming can be generally classified into three categories: 1) centralized [1], [2], [6], 2) flooding-based [7], [8], and 3) DHT-based [3], [4], [9], [10]. Obviously, the centralized method incurs a single point of failure (i.e., the central server) and thus is neither scalable nor resilient. Besides, the flooding-based method lacks scalability because of the exponentially increasing communication cost ($\propto d^{TTL}$, where d is the node degree and TTL is the number of flooding hops). In this section, we mainly focus on the complicated DHT-based method.

The DHT-based lookup algorithms rely on an underlying DHT network, where peers publish the hash value of their sharing data to the closest peer. In the P2P-VoD system PROP [3], when a peer gets a video block, it publishes the information of the block to the DHT. If another peer wants to fetch a block, it searches in the DHT for a supplier. This approach brings two problems: 1) continuously publishing data update information brings considerable communication overhead; 2) when a peer moves on and discards some blocks from its buffer, it needs to send deleting messages to update the DHT. Yiu et al. proposed VMesh [4] which integrates three mechanisms: scheduling, storage, and lookup. For the lookup service, media content is divided into large segments (up to 5-minute video), and a segment is published after completely downloaded. However, which segment should be stored depends on a complicated data cache and replacement mechanism. Recently, OBN [9] introduced the idea of attribute lookup. OBN considers the DHT update problem by using the buffer relationship between peers. Nevertheless, how to efficiently compute the play point distance between peers is not discussed by OBN. In a word, the DHT-based method is efficient but complicated and not resilient. In comparison, our proposed RELookup algorithm utilizes the design of "play point distance" to provide efficient lookup service, and the non-DHT (super node-based) overlay is simple and resilient.

III. System Model

By using a bootstrap server (note that almost every P2P system requires a bootstrap server for some initialization works like adopting newly joining nodes), super nodes are picked out from peers and assigned with a range of data blocks to manage. When a new super node is picked, it gets a list of several existing super nodes from the bootstrap server as its neighbors. Every super node maintains a *global information table* which is used to index the peers who possess corresponding data blocks. Here the key problem is how to design the global information table for fast VoD lookup service and low communication overhead, which will be addressed in detail in Section IV-B.

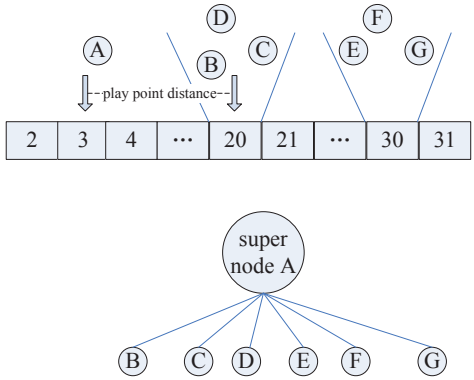

Figure 1. A super node A and its affiliated common peers.

Block. A video is segmented into multiple blocks. Each block corresponds to a play point.

Super Node. Super node plays an important role in RELookup by both reducing the communication overhead and providing fast VoD lookup service. On one hand, it acts as a common peer who just views the video; on the other hand, it organizes its affiliated common peers (like a "network hub") and provides VoD lookup service for other peers. How to select a super node from peers will be presented in Section IV-A. Figure 1 illustrates a super node A and its affiliated common peers (from B to G). As a common peer, it is playing the block 3; as a super node, it manages the index of the blocks between 20 and 30. Some common peers (from B to G) who are playing a block between 20 and 30 are recorded by A, and they send periodical heartbeat messages to A. These common peers are viewed as *child nodes* of A.

Play point distance. As to a super node, the playing blocks of its child nodes change over time, and the block index of the super node changes accordingly. We use the function *F(t)* to describe how the block index of a super node changes with time. Suppose one block corresponds to one-minute video, then a simple example of $F(t)$ is:

$$F(t) = \begin{cases} 0, & t \in (0,1] \text{ minutes;} \\ 1, & t \in (1,2] \text{ minutes;} \\ \cdots & \\ \text{n}, & t \in (n, n+1] \text{ minutes.} \end{cases}$$

More specifically, we define the block index of a super node as $B = \{b_1, b_2, \cdots, b_n\}$. Then B changes with time like: $B(t) = B(0) + F(t)$, which means for each $b_i \in B$, $b_i(t) = b_i(0) + F(t)$. Obviously, if the current playing block of a child peer is c and $c \in B$, then after t minutes,

the child peer's playing block will be $c+F(t)$, where $c+F(t) \in B(t)$. In Figure 1, the super node A is playing the block 3, and is managing the index of the blocks between 20 and 30. One minute later, A will be playing block 4, and A will manage the index of the blocks between 21 and 31.

Super Node Overlay. Super nodes are loosely organized into an unstructured and connected overlay. The network connectivity is maintained by the bootstrap server. In the unstructured super-node overlay, every super node maintains a *global information table* to record the block index information of other super nodes. Ideally, the global information table should be consistent in every super node. When a new super node is picked, it gets a list of several existing super nodes from the bootstrap server as its neighbors, and it uses its neighbors' global information tables to construct its own global information table. Each super node periodically sends heartbeat messages to its neighbors, where its global information table is contained in the heartbeat message for other super nodes' reference, so that the global information tables of super nodes will gradually converge to a consistent state.

IV. RELookup Design

A. Selecting Super Nodes

In RELookup, super nodes take charge of both data block index and overlay organization. When a super node logouts from the system or a super node decides to alleviate its load, it has to transfer the corresponding block index and child nodes to another super node. This process will bring about considerable communication cost, so it is necessary to select a highly stable peer to act as a super node. The metric *OLR* (online ratio) is used to evaluate how stable a peer is. If a peer joins and leaves the system for n times in a *period* (usually 24 hours), in each online interval (join-leave) it remains up for t_i time, then *OLR* is defined as: $OLR = \frac{\sum_{i=1}^{n} t_i}{period}$.

Each peer records its online history and thus its *OLR* can be easily calculated. Each time when a peer joins the system, it uploads its latest *OLR* value to the bootstrap server. When a new super node is required, the bootstrap server selects the online peer with the highest *OLR* as the new super node.

B. Flooding Avoidance and GIT

In RELookup, the GIT (*global information table*) is used to lookup for candidate data suppliers of a block. Traditionally, to maintain a GIT which can reflect the in-time state of every super node, super nodes have to flood state update messages continuously which incurs enormous communication cost. As to our proposed RELookup algorithm, "play point distance" (see Section III) is combined with GIT to avoid continuous flooding messages. In particular, every super node executes at most three flooding operations (at the time of join, split, and logout, respectively) in its working life.

Figure 2. Attributes of a flooding message.

When a new super node is selected, the first message is flooded to the system to notify its join. It contains the super node's block index. Since a super node's block index changes with time, traditional lookup algorithms have to reflood the message and then other super nodes update their state. As to RELookup, by using play point distance to record the super node's block index, super nodes do not need to reflood such a "join" message.

The second message is flooded after a super node has performed the split process (i.e., a super node is split into two super nodes to manage the original block index. Detailed description is contained in Section IV-D). This message informs other super nodes that this super node has been "split" and then will not accept any peer as its child nodes. A super node usually performs the split process at most once. The third message is the logout message which is flooded when a super node log outs. On receiving this message, other super nodes remove this super node's record from their GITs.

As in Figure 2, the flooding message just contains four attributes: $m.ip$ (node ip address), $m.playingblock$ (the playing block of the super node), $m.blocks$ (the block index of the super node) and $m.split$ (the split flag of the super node). Once a super node gets such a message, it formats this message into its GIT. We denote an item in the GIT as g, which has three attributes: $g.ip$ (the ip address), $g.blocks$ (the block index of the super node sending this message), and $g.split$ (the split flag). The pseudo codes of how a message is formatted into a GIT goes as in Algorithm 1. Line 4 should be paid more attention to because here play point distance is utilized to record the block index of a super node.

Algorithm 1 Message formatting

```
1: g.ip = m.ip
2: g.split = m.split
3: for each block in m.blocks do
4:     g.blocks = m.blocks - g.playingblock;
5: end for
6: record g in the global information table
```

C. Identifying the Block Index of Super Nodes

As mentioned above, the block index of a super node changes with time and other super nodes can be aware of this without communication. An example below is used to explain how this change can be aware of without communication.

Considering a super node A whose playing block is b_a and block index is denoted by BL. The super node A floods the message (b_a, BL), and then another super node B whose playing block is b_b receives this message and records this message in its global information table as (A, $BL - b_b$). After t minutes, the playing block of A becomes $b_a+F(t)$ and the block index is $BL+F(t)$. Under the usual assumption that every peer possesses the same playback rate, the playing block of B becomes $b_b + F(t)$, and thus to get the latest block index of A, B just needs to locally calculate $(BL - b_b) + (b_b + F(t))$.

In a real system, we can not ensure that every super node starts a block within some pre-defined time limit. So the record may be not accurate. However, if the search scope is extended a little, we can still get the required data suppliers. Suppose A enters the system at t_a and gets a block r_a as its playing block, and the block index BL is returned to it by a super node. A records it as $BL - r_a$. Then A floods this message and B receives it. Now the playing block of B is r_b which starts at the time t_b. Since A's message has been flooded, B knows the block index of A is BL and records it as $BL-r_b$. Then at time t, the block index $BL(t)$ calculated by A is: $r_a+F(t-t_a)+BL-r_a = F(t-t_a)+BL$, and the block index $BL'(t)$ calculated by B is: $r_b + F(t - t_b) + BL - r_b = F(t - t_b) + BL$. Using $F(t-t_a) - F(t-t_b)$ to evaluate the error, it can be easily proved that $|t_a + \Delta t - t_b| < 1$ minute (suppose one block corresponds to one-minute video). Δt is the time delay for the message to be transferred from A to B. As a result, we regard it as far less than 1 minute, and then we get $|t_a - t_b| < 1$ minute. Assuming $t_a > t_b$ and $t_b = t_a - \alpha$, where $0 < \alpha < 1$, we have

$$F(t - t_a) - F(t - t_b) = F(t - t_a) - F(t - t_a + \alpha);$$

If $\lfloor t - t_b \rfloor = \lfloor t - t_a + \alpha \rfloor$ ($\lfloor x \rfloor$ denotes the floor of x), then $F(t - t_a) - F(t - t_a + \alpha) = 0$; otherwise, $F(t - t_a) - F(t - t_a + \alpha) = 1$. Through this case we see that if we want to find a block $c \in BL$, we just need to find the blocks c, $c - 1$, $c + 1$.

D. Scalability

When the scale of the P2P-VoD system grows, a super node may find itself cannot accept any new child node. Then, it asks the bootstrap server for splitting into two new super nodes. The block index of each new super node is a copy of a half of the block index of the original super node, respectively. Then the original super node's split flag is set to true. At this time, we have three super nodes. Now a newly joining peer would find that there are two super nodes taking charge of its playing block, so it takes the one whose split flag is false as its super node. As time goes by, the original super node would have no child and then logout.

When the scale of the P2P-VoD system shrinks, we have to merge the block index of super nodes. Because each block index changes with time, we can hardly record them. Therefore, a super node simply leaves the system and does nothing to its block index. Obviously, its block index will be gradually forgotten by the system. When a new node wants to join the system and finds that there is no super node taking charge of its playing block, it will remind the bootstrap server of such a block and thus the bootstrap server will ask the super node who has the fewest child nodes to take charge of this block.

V. Performance Evaluation

A. Methodology

Real-trace driven simulations are executed to evaluate the performance of the RELookup algorithm. The traces contain 30 real overlay topologies whose data was collected from http://dss.clip2.com. The data contains each nodes ID, IP, port, ping time (from a central node), bandwidth and so on. The trace topologies scale from 100 to 10000 nodes, with an average node degree from less than 1 to 3.5. Because the average node degree is too small for media streaming, we add random edges into the overlay to let every node hold $M = 5$ connected neighbors. According to our simulation experience, $M = 5$ is usually a good practical choice and using a larger M cannot bring more benefit. In each simulation, we choose one real-trace topology as the initial overlay topology, and then for every 20 ms, we add a new peer into the system to evaluate the system scalability. Meanwhile, for every second, we perform node churn operations by selecting 5% of the existing peers to leave the system to evaluate the system resilience. To simulate the fluctuation of the bandwidths of peers, we assign a random value of bandwidth between 400 Kbps and 500 Kbps to each peer. We have performed simulations on all the 30 real overlay topologies and presented the results in typical system scales. Besides, we compare the evaluation results of RELookup with the flooding-based method, since the flooding-based method has the most similar overlay organization as RELookup (compared with the centralized method and the DHT-based method).

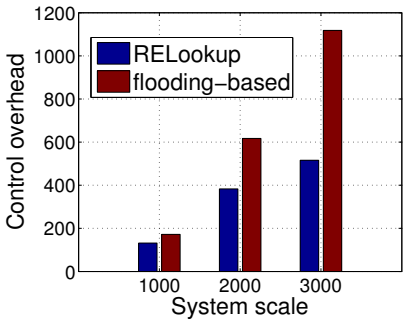

Figure 3. Control overhead.

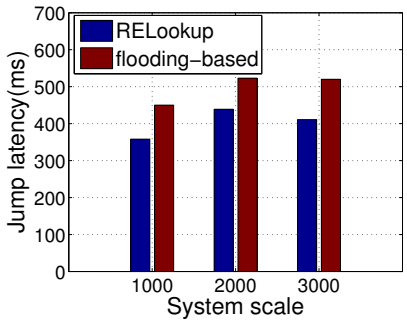

Figure 4. Jump delay.

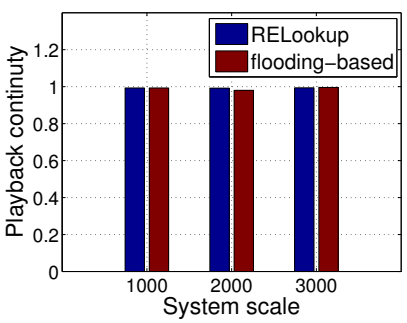

Figure 5. Playback continuity.

B. Metrics

- *Control overhead*. Control messages include the flooding messages sent from super nodes and the heartbeat messages between neighboring peers/super nodes.
- *Jump delay* denotes how long a peer needs to find the candidate data suppliers when the peer wants to jump on the video.
- *Playback continuity* denotes the ratio of blocks that arrive before or on the playback deadline.
- *Number of super nodes*. Because super nodes play an important role in the RELookup algorithm, we record the evolution of the number of super nodes as the system scales. It reflects how our algorithm adapts to the system scaling.
- *Accuracy of GIT*. In RELookup, the global information table (GIT) is used to find candidate data suppliers of a block. Thus, the accuracy of the global information table is tightly related to the efficiency of VoD lookup service. Specifically, the accuracy of the global information table is defined as the ration of accurate records over all records in the global information table.

C. Evaluation Results

Control overhead. As shown in Figure 3, the control overhead of RELookup is much lower than that of the flooding-based method. In addition, we discover that the control overhead approximately increases linearly, because most of the control overhead is caused by periodical heartbeat messages.

Jump delay. From Figure 4, we discover that the jump delay of RELookup is lower than that of the flooding-based method. As to the flooding-based method, the peer who has sent the lookup request should wait until all data suppliers respond, which is the main reason why the flooding-based method takes a longer time to jump to a new video position.

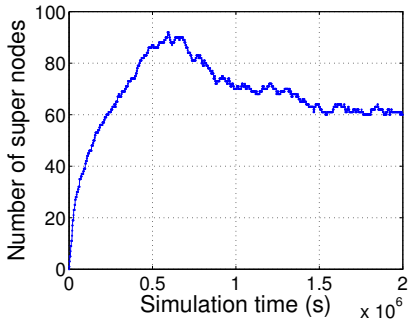

Figure 6. Number of super nodes.

Playback continuity. We track the playback continuity of RELookup and the flooding-based method with different system scales, as depicted in Figure 5. Both methods can keep the playback continuity very close to 1.0.

Number of super nodes. Figure 6 plots the evolution of the number of super nodes in RELookup. At first, the number of super nodes increases as the system scale grows from 0 to 2000 because super nodes continuously split into more super nodes during the first stage. Then the number of super nodes gradually decreases until stable. The decreasing results from old super nodes' leaving the system (Recall that old super nodes do not leave the system instantly after they split). When old super nodes have all left, the number of super nodes keeps stable.

Accuracy of GIT. From Figure 7, we see that the accuracy of GIT is very high (always above 0.99). We also discover that the accuracy curve has frequent churns which mainly results from new super nodes' joining or old super nodes' leaving the system.

VI. Conclusion

In this paper we investigate how to design an efficient and resilient lookup algorithm for P2P-VoD streaming to

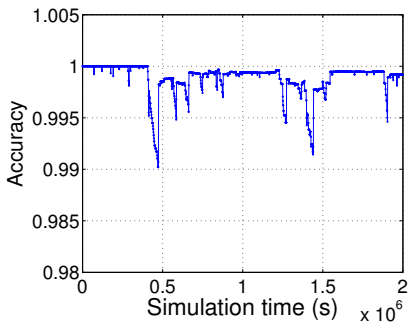

Figure 7. Accuracy of GIT in RELookup.

provide effective operation of random jump on the video. After analyzing the pros and cons of the state-of-the-art lookup algorithms including the centralized, flooding-based, and DHT-based methods, we propose RELookup, a novel lookup algorithm which places peers on a resilient super node-based overlay and meanwhile utilizes the design of play point distance to efficiently locate candidate data suppliers. Furthermore, deliberate measures, i.e., special design of the message format and the node state, have been taken to reduce the coordination costs between super nodes to very little. The results of trace-driven simulations confirm the effectiveness of RELookup.

Acknowledgement

The work is partly supported by the China NSF grants (60825205, 61073152, 61021062, 61133006, 61073015) and the China "973" grant 2011CB302305.

References

[1] Y. Guo, K. Suh, J. Kurose, and D. Towsley. "P2Cast: peer-to-peer patching scheme for VoD service," In WWW, 2003.

[2] T. Do, K. Hua, and M.A. Tantaoui. "P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment," In IEEE ICC, 2004.

[3] L. Guo, S. Chen, S. Ren, X. Chen, and S. Jiang. "PROP: a Scalable and Reliable P2P Assisted Proxy Streaming System," In IEEE ICDCS, 2004.

[4] W. Yiu, X. Jin, and S. Chan. "VMesh: Distributed segment storage for peer-to-peer interactive video streaming," IEEE journal on selected areas in communications, vol. 25, no. 9, 2007.

[5] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang. "Challenges, design and analysis of a large-scale p2p-vod system," In ACM SIGCOMM, 2008.

[6] X. Yang, M. Gjoka, P. Chhabra, A. Markopoulou, and P. Rodriguez. "Kangaroo: Video seeking in P2P systems," In IPTPS, 2009.

[7] X. Jiang, Y. Dong, D. Xu, and B. Bhargava. "GnuStream: A P2P Media Streaming System Prototype," In IEEE ICME, 2003.

[8] J. Li. "PeerStreaming: A practical receiver-driven peer-to-peer media streaming system," MSR-TR-2004-101, 2004.

[9] C. Liao, W. Sun, C. King, and H. Hsiao. "OBN: Peering for Finding Suppliers in P2P On-demand Streaming Systems," In IEEE ICPADS, 2006.

[10] T. Zhang, X. Cheng, J. Lv, Z. Li, and W. Shi. "Providing Hierarchical Lookup Service for P2P-VoD Systems," To appear at ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP), 2011.

[11] I. Stoica, et al. "Chord: A scalable peer-to-peer lookup service for internet applications," In ACM SIGCOMM, 2001.

[12] B. Zhao, and et al. "Tapestry: A Resilient Global-Scale Overlay for Service Deployment," IEEE Journal on Selected Areas in Communications, vol. 22, 2004, pp. 41 - 53.

[13] M. Ripeanu. "Peer-to-peer architecture case study: Gnutella network," In IEEE P2P, 2001.

[14] http://en.wikipedia.org/wiki/EDonkey_network.

[15] http://www.kazaa.com.