An Overlapping Structured P2P for REIK Overlay Network

Wenjun Liu, Jingjing Song, Jiguo Yu

City College, Wenzhou University, Wenzhou, Zhejiang, P. R. China
Email: w-jliu@163.com

The Second Artillery NCO School, Qingzhou, Shandong, P. R. China
Email: jjs54@126.com

School of Computer Science, Qufu Normal University, Rizhao, Shandong, P. R. China
Email: jiguoyu@sina.com

Abstract

REIK is based on a ring which embedded an inverse Kautz digraph, to enable multi-path P2P routing. It has the constant degree and the logarithmic diameter DHT scheme with constant congestion and Byzantine fault tolerance. However, REIK did not consider the interconnection of many independent smaller networks. In this paper, we propose a new approach to build overlay network, OLS-REIK which is an overlapping structured P2P for REIK overlay network. It is a more flexible interconnecting different REIK network. Peers can belong to several rings, allowing this interconnection. By connecting smaller structured overlay networks in an unstructured way, it provides a cost effective alternative to hierarchical structured P2P systems requiring costly merging. Routing of lookup messages is performed as in REIK within one ring, but a peer belonging to several rings forwards the request to the different rings it belongs to. Furthermore a small number of across point is enough to ensure a high exhaustiveness level.

1. Introduction

Peer-to-peer (P2P) system has emerged as a powerful paradigm for exchanging data. They are used by millions of users for file sharing over the Internet. Typically, P2P networks are envisioned to find to a broad range of applications, moving way beyond their current applications as infrastructure for file sharing and exchange such as in Gnutella or Morpheus. For example, P2P networks for Semantic Web Services can be used to provide distributed access to these services without requiring a central service directory.

Unstructured peer-to-peer networks offer no guarantee on the diameter because nodes interconnect in a random manner, resulting in anything other than an efficient topology. These unstructured systems are
often criticized for their lack of scalability [3], which can lead to partitions in the network resulting in small islands of interconnected nodes that cannot reach each other. However, these same random connections offer the network a high degree of resiliency where the operation of the resulting network as a whole is tolerable to node removal and failure. In contrast, structured P2P networks based on distributed hashing tables (DHTs), such as Chord, CAN, Pastry and Tapestry have been designed to provide a bound on the diameter of the system, and as a result, on the response time for nodes to perform queries. However, these systems impose a relatively rigid structure on the overlay network, which is often the cause of degraded performance during node removals, requiring non-trivial node maintenance. This results in certain vulnerabilities that attackers can target and exploit. Due to the design of DHTs, these structured topologies are also limited in providing applications with the flexibility of generic keyword searches because DHTs rely extensively on hashing the keys associated with objects [1].

There have also been several academic P2P system proposals that offer very efficient key lookups. These systems are based on implementing a distributed hash table (DHT) over these nodes and they impose a certain structure on the overlay network constructed by the nodes. Any key lookup can typically be resolved by exchanging $O(\log N)$ messages where $N$ is the number of nodes in the system, e.g., Koorde[2], KZCAN[7], BRGK[8] and REIK[5, 9].

Although DHT based P2P systems are praised for their efficiency in search, most existing P2P systems are still based on some simple design. For example, Napster uses one or more centralized servers for indexing objects, while Gnutella and Freenet allow peers to be loosely coupled in a fully distributed but unstructured manner. P2P systems with some centralized index servers are hard to scale and exhibit single point of failure. Unstructured P2P systems are named because they do not deliberately maintain their network in a specific structure other than to keep it connected. As a result, there is much freedom for nodes to choose other nodes as entries in their routing tables. So node joining and leaving poses little problem to the network, and so the systems are highly robust and easily maintained. However, the search costs could be very high as search in an unstructured network is more or less an exhaustive process, and in the worst case the entire network must be traversed. That is, the search cannot be guaranteed in bounded costs.

The original DHT designs tend to treat nodes equally, and thus emphasize load balancing among them. However, empirical studies have shown that diversity exists among participating nodes in P2P systems. For example, Saroiu et al. [4] observed that 50% of Gnutella nodes have session time less than 60 min, and that 10% of nodes are relatively more stable than the others. Indeed, existing P2P systems like Morpheus, KaZaA, and eMule have all taken heterogeneity into account and employed super peers that are more powerful than ordinary peers to act as regional centralized index servers. Other systems, e.g., Brocade [10] and Expressway [6], use network access points as super peers to help routing as they are often very stable and with high bandwidth.

In this paper, we propose a new approach to build overlay network, OLS-REIK which is an overlapping structured P2P for REIK overlay network. It is a more flexible interconnecting different REIK network. Peers can belong to several rings, allowing this interconnection. By connecting smaller structured overlay networks in an unstructured way, it provides a cost effective alternative to hierarchical structured P2P systems requiring costly merging. Routing of lookup messages is performed as in REIK within one ring, but a peer belonging to several rings forwards the request to the different rings it belongs to. Furthermore a small number of across point is enough to ensure a high exhaustiveness level.

In the following, we firstly describe the REIK network in section 2. In section 3, we specify the OLS-REIK and its performances. Finally, conclusions are discussed in section 4.

2.REIK

REIK uses the ring which embedded an inverse Kautz digraph as its topology. Like Chord, REIK uses consistent hashing to map keys to nodes. It is a distributed lookup protocol that provides the following basic operation in a P2P network: maps a given key onto a node. Data location can then be implemented by assigning a key with each data item, and storing the key at the node which the key maps to. Node
identifiers are obtained by hashing their IPs. A key $k$ is assigned to the successor node of key $k$, $\text{successor}(k)$.

To represent a real-life P2P overlay network, REIK system contains only a few of the possible $d^m + d^{m-1}$ nodes, where $d$ and $m$ are the degree and diameter of the embedding inverse Kautz digraph, respectively. Thus, some points on the identifier circle correspond to nodes that have joined the system, while many points on the ring correspond to imaginary nodes. To ensure that queries of object keys can be resolved in the ring, every node maintains the link to its successor in the ring. So the successor node of key $k$ can be determined by traversing through the successor links until the node $\text{successor}(k)$ is reached. To speed up the search process, each node also maintains a finger table, where every finger points to an inverse Kautz node in the identifier ring. See Fig.1 for illustration.

A node $u$ joins the REIK network by contacting any existing node $v$, and asks $v$ to find $u$‘s successor in the ring. Then $u$ is integrated into the REIK ring by setting it successor link to point to the successor, and then builds its finger table with the help of the successor. After new node $u$ joins the system, some nodes have to update their fingers originally pointing to $u$‘s successor now to $u$. In REIK, this is solved by letting each node periodically execute fix fingers to keep fingers up to date. Note that in the above joining process, after $u$ has located its successor, only the successor link of $u$ has been set. The predecessor links of $u$ and the successor as well as the successor link of $u$‘s predecessor are set by letting nodes periodically execute the stabilization process. This process also allows existing nodes to learn new nodes that have concurrently joined the system.

When a node leaves or fails, its departure must be detected to maintain the connection of the ring. This is done by letting processes periodically check their predecessors and successors. In addition, to enhance the robustness of the system, each node keeps a successor list, rather than single successor node.

The correctness of lookup relies on the correctness of the successor links, while the efficiency of lookup relies on the correctness of the finger tables. This is because when a node forwards a lookup message to its $i^{th}$ finger node but the finger node has failed, the lookup message will be re-transmitted to its $(i-1)^{th}$ finger node, and so on until the $0^{th}$ finger, i.e., the successor link, has been reached. So the maintenance of finger tables accounts for most of the overhead. The maintenance of successor links is crucial to the correctness of lookup function, while the maintenance of routing table affects only lookup efficiency.

To enhance the ability of tolerating Byzantine faults, the REIK uses the multiple disjoint routes. Two routes are overlapped, if they contain some intermediate nodes in common. The route overlapping approaches zero as the node degree of REIK overlays becomes very large. To achieve higher fault tolerance, the route overlapping should be made as low as possible. By definition, two overlapped routes must have at least one intermediate node in common. Byzantine fault is formed with collusive nodes. In real-life overlay networks, nodes are sparsely distributed in a large key space. Thus REIK overlay with incomplete inverse Kautz achieves Byzantine fault tolerance.

Just like finger pointers in Chord, REIK’s inverse Kautz pointer is merely an important performance optimization, a query can always reach its destination slowly by following successors. Because of this property, REIK can use Chord’s join algorithm. Similarly, to keep the ring connected in the presence of
nodes that leave, REIK can also use Chord’s successor list and stabilization algorithm. Hence, REIK is a purely decentralized structured P2P overlay network that provides mechanisms for efficient localization of the node that stores a particular data object. It is a message driven dynamic structure that is able to adapt as nodes join or leave the system.

3. OLS-REIK

We here discuss the structure and properties of OLS-REIK. OLS-REIK is a more flexible interconnecting different REIK network. Peers can belong to several rings, allowing this interconnection. By connecting smaller structured overlay networks in an unstructured way, it provides a cost effective alternative to hierarchical structured P2P systems requiring costly merging.

3.1. Overall description of OLS-REIK

OLS-REIK is an overlapping structured P2P for REIK overlay network. Peers can belong to several distinct rings. A peer wishing to join a ring comes with a list of resources it offers to this ring’s community. The more relevant resources the peer injects into the ring, the higher the probability to successfully enter it is. It is clear that the more rings the peer is registered to, the larger its routing table will be. Nevertheless, we can assume that the numbers of rings a peer belongs to will be pretty low. Moreover it is the peer’s choice to belong to more floors, thus it knows it has the capacity to deal with a routing and storage overhead. Each ring has a proper hash function in order to perform consistent hashing of peers and keys within it. Peer variables used to perform routing, like its predecessor on the ring, its successor on the ring and the entries of its routing table, must be upgraded in order to take into account the multi-ring extension. The Fig. 2 shows a topology made of two rings for OLS-REIK.

In order to avoid lookup cycles when doing cross-rings search, each peer maintains a list of already processed requests flag in order to discard previously seen queries. Also, each lookup process has a Time-To-Live value which is decreased each time we cross the boundaries of the ring. These two features prevent the network from generating unnecessary queries and thus reducing the global OLS-REIK number of messages. A good property of the routing mechanisms is that with a fairly low amount of across points, we can still achieve pretty high query exhaustiveness.

When a peer looks up a resource on a given ring using the hash function of the ring, OLS-REIK routing is launched. A unique flag identifier for the query is created. If the routing goes through a across point, then the lookup is forwarded in parallel to all the rings the across point belongs to, otherwise the routing goes on as in a standard REIK ring. To do this, each peer needs to know the hash function of the ring. This means that keys need to be hashed at every ring change. The rationale of this propagation is simple: the more rings you explore, the higher the probability of success will be. It is important to notice that while the search within a single ring lookup is exhaustive and logarithmic in the number of peers, the whole lookup
in OLS-REIK can be non exhaustive with a routing complexity that can vary according to the number of rings times a logarithmic factor.

3.2. Routing mechanism for OLS-REIK

Participation of peers in an overlay network dynamically changes over time. Each peer can freely decide to join or leave an overlay network at any time. To maintain the structural integrity of an overlay network a maintenance strategy is required, which compensates for changes to the network structure due to peers going offline or failure of network connections. In all overlay networks, joining the network is done explicitly by a join operation, whereas leaving typically is implicit as peers may simply go offline or crash or their network connection may drop. Regardless whether peers leave gracefully or not, changes in the participation in an overlay network typically require the application of a maintenance strategy.

We present the details of the algorithms to build OLS-REIK. Let $Rid$ is the identifier of the ring, with a slight abuse of notation, the hash function $hash(Rid)$ used at this ring. Every peer contributes actively to routing through its routing table and to resource exchange.

**Algorithm 1.** The search processing for OLS-REIK.

**Input:** The peer $p$ and the key $k$.

**Output:** The result set $S$.

1. Initialize $f$ is a new unique flag for this lookup.
   1. Send $p.succ$ to the peer $p$;
   2. If $p$ is the aliveness then
   3. Update the hot peer list with $p$ at the ring;
   4. Update the hot ring list with $r$;
   5. Return the set $S$ with offering the key $k$;
   6. Find the $succ$ of $p$;
   7. If $f$ is the process for join a ring;
   8. If the key $k \in (p, p.succ]$ then
   9. Send the $p.succ$ to $p$;
   10. Else
   11. For $i = m$ to 1 step -1
   12. If testing the hand table as in REIK then
   13. Return the finger of the ring;
   14. Return the current peer $p$;
   15. Sent to the next hop;
   16. Else if the lookup is not processed then
   17. Set the $f$ to already processed;
   18. For the current peer $p \in$ all rings
   19. If the key $k \in (p, p.succ]$ then
   20. Send the $p.succ$ to $p$;
   21. Else
   22. Send findsucc to the next hop.

Algorithm 1 shows how the search processing on any node is performed. When the first element of the message is a join flag, it means that the lookup serves a join purpose. The request is then routed as the join of REIK, and corresponds to the routing process of either a resource registration or a peer insertion. If the
first element of the message is a numeric flag, it means that the message is part of a resource lookup request. The message can then be routed in several rings the current peer belongs to. Firstly, the flag of the request is checked. If the request was already processed, it is simply ignored; otherwise, the request flag is saved. When a search is successful, a found message is sent back to the lookup’s initiator. On receipt of found, the initiator checks the aliveness of the peer and updates its hot peer and hot ring lists according to its satisfaction. Finally, it remotely reads the values wanted.

Algorithm 2. The ring join processing for OLS-REIK.

1. Current node invited by q to join the ring r;
2. If the invitation is a “good deal”
3. Add ring r to the hands;
4. Add a succ for ring r to the successor;
5. Add a pred for ring r to the predecessor;
6. Find the succ of q;
7. Receiving the response;
8. For all the resources offered by q
9. Find the node hosting the table entry;
10. Waiting for response;
11. Updated the routing table;
12. The current node ask to q to join the ring r;
13. Accept q at the ring r.

The Algorithm 2 shows the pseudocode for proposing and accepting new connections. Note that the strategy for peers and rings selection can be based on any criteria.

4. Conclusions

Peer-to-peer applications are used to share large volumes of data. In this paper, we propose a new approach to build overlay network, OLS-REIK which is an overlap- ping structured P2P for REIK overlay network. It is a more flexible interconnecting different REIK network, and it aggregates smaller structured overlay networks in an unstructured fashion based on intersection peer. Next step is a complete analysis of such topologies, more simulations, and an actual implementation followed by real deployments of this promising paradigm.

Acknowledgment

The authors would like to thank Promotional Foundation for Middle-aged or Young Scientists of Shandong Province for contract (2005BS01016), SRI of SPED (J07WH05), the scientific research project of city college, Wenzhou University (KYZD200903) under which the present work was possible.

References


