CISS: An Efficient Object Clustering Framework for DHT-Based Peer-to-Peer Applications

Jinwon Lee, Hyonik Lee, Seungwoo Kang, Sungwon Choe, and Junehwa Song

Division of Computer Science,
Department of Electrical Engineering & Computer Science,
Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea
{jcircle, hyonigi, swkang, sungwon, junesong}@nclab.kaist.ac.kr

Abstract. Distributed Hash Tables (DHTs) have been widely adopted in many Internet-scale P2P systems. Emerging P2P applications such as massively multi player online games (MMOGs) and P2P catalog systems frequently update data or issue multi-dimensional range queries, but existing DHT-based P2P systems can not support these applications efficiently due to object declustering. Object declustering can result in significant inefficiencies in data update and multi-dimensional range query routing. In this paper, we propose CISS, a framework that supports efficient object clustering for DHT-based P2P applications. While utilizing DHT as a basic lookup layer, CISS uses a Locality Preserving Function (LPF) instead of a hash function. Thus, CISS achieves a high level of clustering without requiring any changes to existing DHT implementations. Technically, we study LPF encoding function, efficient routing protocols for data updates and multi-dimensional range queries, and cluster-preserving load balancing. We demonstrate the performance benefits of CISS through simulation.

1 Introduction

Distributed Hash Table (DHT)-based overlay networks [15][17][20][21] have recently emerged as a scalable and efficient infrastructure for wide-area data management. DHT is already adopted in many P2P systems including a wide-area file system [5] and an Internet-scale query processor [9]. These P2P systems mainly focus on an environment in which data updates are rare and exact match queries are the norm. However, emerging P2P applications frequently update data or issue range queries. For example, MMOGs intensively generate streams of updates such as players' locations and status [3][10]; P2P catalog systems intensively issue multi-dimensional range queries such as interest area queries [13].

Existing DHT-based P2P systems [5][9] can not support such applications efficiently due to object declustering. These P2P systems use a hash function to distribute objects randomly across different peer nodes. Thus, while they are effective in achieving a high level of load balancing, objects are totally declustered; even highly correlated objects are spread over different peer nodes. Such object declustering can result in significant inefficiencies in both data update and multi-dimensional range

W.S. Ng et al. (Eds.): DBISP2P 2004, LNCS 3367, pp. 215–229, 2005.

query routing. In data-intensive P2P applications such as MMOGs, DHT lookups have to be performed at every data update even though consecutive data updates are semantically close. Thus, it increases not only the communication overhead of the DHT layer, but also the latency of update routing. However, when objects are clustered, semantically close updates can be routed to the same peer node without having to perform additional lookups. In multi-dimensional range query-intensive P2P application such as P2P catalog systems, queries search for semantically related objects. Thus, when totally declustered, each key value in a query range should be enumerated and individually searched for via a separate DHT lookup. However, when objects are clustered, multiple key values can be searched for via a single lookup. Thus, the number of DHT lookups needed for query processing can be greatly reduced.

In this paper, we propose CISS (Cooperative Information Sharing System), a framework that supports efficient object clustering for DHT-based P2P applications. While utilizing DHT as a basic lookup layer, CISS uses a Locality Preserving Function (LPF) instead of a hash function. Thus, CISS achieves a high level of object clustering without requiring any changes to existing DHT implementations. Consequently, CISS significantly reduces the number of DHT lookups needed for data updates and multi-dimensional range queries.

In realizing CISS, there are three technical issues that must be taken into consideration. *First, CISS has to construct an N-bit key from multiple attributes for each object while preserving locality.* In order to preserve locality, the keys of two objects should be similar if attribute values of those objects are semantically related. The LPF is responsible for this key encoding. The LPF first encodes each attribute value to a shorter-length bit key. It then maps multiple such shorter-length bit keys to a one-dimensional *N*-bit key using the Hilbert SFC. Since the data types of each attribute can be diverse in practice, the LPF needs to be able to encode the attributes of various data types. We describe this encoding scheme in Section 4.1 with practical examples.

Second, CISS must support efficient routing protocols for data updates and multi-dimensional range queries in order to maximize the benefits of object clustering. To route data updates efficiently, we propose a caching-based update routing protocol. This routing protocol does not perform additional lookups if streams of updates belong to the key range of the most-recently-searched peer node. Each peer node manages semantically related objects and data updates are also usually semantically close. Thus, streams of updates belong to the same node with high probability. To route multi-dimensional range queries efficiently, we propose a forwarding-based query routing protocol that performs a minimal number of costly DHT lookups. In addition, our query routing protocol prevents query congestion.

Third, CISS must perform load balancing while preserving object clustering. Since each peer node in CISS manages semantically related objects, a skewed distribution of objects and queries results in significant load imbalance. To prevent hotspots, load balancing must be performed. However, load balancing mechanisms in existing DHT-based systems such as virtual servers [4][5][14][20] destroy the object clustering property. When using virtual servers, physical peer nodes can manage non-contiguous key ranges, *i.e.* multiple virtual servers. In order to preserve object clustering, physical peer nodes must manage contiguous key ranges. We propose two novel

load balancing schemes, local-handover and global-handover, which preserve object clustering even after load balancing is achieved.

The rest of the paper is organized as follows. Section 2 reviews related work in the area of object clustering in P2P overlay networks. In Section 3, we describe the architecture of CISS. In Section 4, we explain technical issues faced in realizing CISS, including LPF, data and query routing protocols and cluster-preserving load balancing. Section 5 presents results from simulation studies of CISS. Finally, Section 6 concludes with a discussion of our plans for future work.

2 Related Work

In existing DHT-based P2P systems [5][9], exact matching queries are efficiently processed in $O(\log S)$ time, where S is the number of nodes in the P2P overlay network. However, streams of data updates and multi-dimensional range queries are not supported well due to object declustering in such systems. Recent research has focused on alleviating these shortcomings.

Much of this research [1][7][11][18] attempts to provide simple one-dimensional range queries over P2P overlay networks. In [1][18], the authors extend CAN [15] for range queries by utilizing query flooding techniques. In [7][11], they propose newly designed range addressable P2P frameworks which are not compatible with existing DHT implementations.

CLASH [12] and PHT [16] apply an extensible hashing technique to DHTs. They efficiently achieve an adaptive object clustering as well as support range queries. Due to the need for depth searching, an exact match lookup takes $O(\log(D)\cdot\log(S))$ time, where D is the maximum depth of the key and S is the number of nodes. However, multi-dimensional range queries have not been considered yet in these research projects.

Squid [19] supports multi-dimensional range queries over DHTs by using the Hilbert Space Filling Curve (SFC). Recursive refinement of queries in Squid significantly improves the performance of query routing, but it can incur query congestion. Thus, the overall scalability of a DHT-based P2P system is limited.

From the standpoint of real systems, much of this previous research did not consider several critical technical issues. First, it is not clear how to encode real attribute values to N-bit routing keys. In this paper, we clearly describe such an encoding scheme with practical examples. Second, even though many previous works focused on query routing, in fact it is data updates that are the major performance bottleneck of data-intensive P2P applications. We propose an efficient update routing protocol to address this. Finally, cluster-preserving load balancing has not been considered yet in those previous works. Load balancing is essential for such P2P systems to be able to work under real environments. However, the benefits of object clustering can be destroyed if we directly apply previous load balancing schemes [4][5][14][20]. Our cluster-preserving load balancing schemes are novel in that sense.

3 System Architecture

CISS is designed for a three-tier P2P system as shown in Figure 1. Such three-tier architecture is similar to existing DHT-based P2P systems [5][9]. While CISS uses DHT as a basic lookup layer by using DHT interfaces, P2P applications utilize CISS as an Internet-scale data management system. For data updates and queries, an interface using a simple conjunctive normal form language is provided to applications (see Table 1). CISS, like common P2P systems, consists of client and server modules. The client module of CISS receives data updates or queries from P2P applications. It then routes them to rendezvous peer nodes for processing. Before routing them, the client module leverages an LPF to encode multiple attributes of an object to an *N*-bit routing key. This key is used to perform a DHT lookup to search for rendezvous peer nodes in the P2P overlay network. The server module of CISS stores data to its repository and processes queries. It then returns matched results to requesting peer nodes. The load balancer in the server module is responsible for cluster-preserving load balancing.

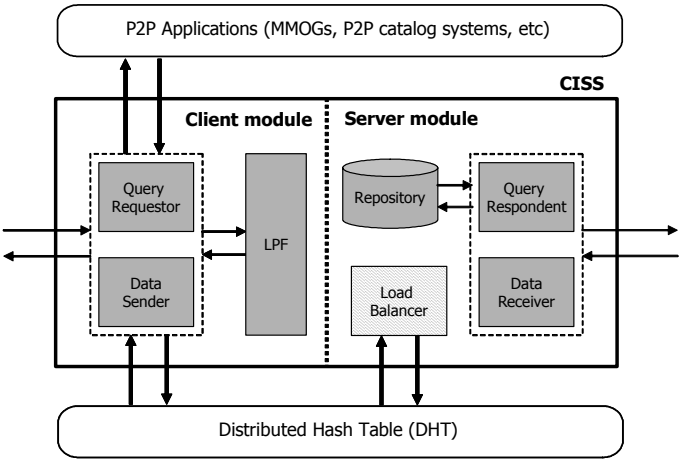
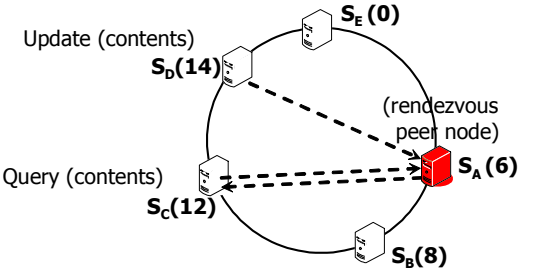


Fig. 1. CISS Architecture

Table 1. Interfaces for DHT and CISS

DHT	CISS
Lookup(key) → IP address	Update: (A1= value) ∧ (A2=value) ∧….
Join ()	Query: Predicate_{A1} ∧ Predicate_{A2} ∧….
Leave()	Predicate = Attribute **Operator** Value Operators = {>, <, ≥, ≤, =}



Peer node	Key range
S_A	(0, 6]
S_B	(6, 8]
S_C	(8, 12]
S_D	(12, 14]
S_E	(14, 0]

Fig. 2. Chord DHT

The scalable and robust nature of CISS stems primarily from utilizing DHT. We explain CISS using Chord [20] as an example DHT environment though any DHT implementation could be used. DHT [15][17][20][21] organizes highly distributed and loosely coupled peer nodes into an overlay network for storing and querying a massive number of objects. In a DHT environment, not only the placement of data objects on nodes, but also the join and leave of nodes in the overlay network can be done efficiently without any global knowledge. As shown in Figure 2, five peer nodes cooperatively manage an *N*-bit key space, where *N* is 4. Each node has a unique node identifier and is responsible for the key range between itself and its predecessor node.

From a database point of view, there are two attractive characteristics for using a DHT-based overlay network.

First, content-based searching in peer to peer networks – DHT makes it possible to implement content-based search networks. Multiple attributes of an object are encoded to form an *N*-bit key which is used to update and locate that object. Data updates and exact match queries which are encoded to the same *N*-bit key are routed to the same rendezvous peer node. After query processing, the matching results are returned to the querying nodes. This rendezvous point approach achieves content-based searching effectively by avoiding query flooding.

Second, efficient wide-area data indexing – It is critical that searches for the rendezvous peer node responsible for a given *N*-bit key should be done efficiently. DHT theoretically ensures that any peer node can look up any object using that object's *N*-bit key in $O(\log S)$ time, where *S* is the number of peer nodes in the overlay network. Lookups proceed in a multi-hop fashion; each node maintains information (IP addresses) about a small number of other nodes (neighbors) and forwards the lookup message recursively to the neighbor that is nearest to the *N*-bit key of the object.

4 Technical Issues

In this section, we describe three technical issues and novel solution approaches in realizing CISS. Specifically, the LPF encoding function, efficient routing protocols for data updates and multi-dimensional range queries and cluster-preserving load balancing are examined.

4.1 Locality Preserving Function (LPF)

The LPF constructs *N*-bit keys of objects while preserving locality. As shown below, this encoding is done in two steps.

Step1: {(A1 = value) ∧ (A2 = value) ∧….} ⟶ {$bits_{A1}$ ∧ $bits_{A2}$ ∧….}
Step2: {$bits_{A1}$ ∧ $bits_{A2}$ ∧….} ⟶ ***N*-bit key of object**

The LPF first encodes each attribute value to a smaller-sized bit key. It then maps multiple bit keys to a one-dimensional *N*-bit key by using the Hilbert SFC. Both steps preserve the locality of objects. Each attribute is encoded to N / D (= M) bits if D attributes are used for key encoding. As a practical value, we can use $N = 160$ and $D = 2$. Thus, $M = 80$. We select $N = 160$ to be compatible with Chord [20] implementation which uses 160-bit key. Also, $D = 2$ because two-dimensional range queries are dominantly issued in both MMOGs and P2P catalog systems. If an attribute of an object is not encoded as part of an *N*-bit key, queries on this attribute must be routed to all nodes in the P2P overlay networks. Thus, all attributes referred to in dominantly issued queries must be encoded to bit keys in order to avoid query flooding. We describe the technical details of each step as follows.

Step1: Bit key encoding of each attribute while preserving locality – LPF classifies data types of attributes into `Numerical` and `String` types, and applies different encoding schemes accordingly. We explain each encoding scheme with practical examples. The encoding scheme for the `Numerical` type handles `int`, `long`, `float`, `double` and `DATE` data-types.

For instance, MMOG (see Figure 3) use `Numerical` attributes. Players are the objects, and their x and y coordinates are the object attributes. In order to preserve locality, each attribute value is simply rescaled by multiplying a coefficient, 2^M / *(Maximum of attribute value)*. For example, {x=60 ∧ y=70} where the maximum of each attribute value is 100 is encoded to {x=1010 ∧ y=1011} if M is four. In the same way, a two-dimensional range query in Figure 3 is also encoded to {(0101 < x < 1110) ∧ (0011 < y < 1110)}. Therefore, objects will be clustered well if the positions of the objects are similar.

For the `String` type, we propose a *hash-concatenation encoding scheme*. P2P catalog systems (see Figure 4) use `String` attributes. Catalogs are the objects and categorized by two attributes, location and product. Each attribute value is represented using a hierarchical naming structure. For example,

A1: location = USA.New York.White Plains.79 North Broadway
("USA" is the value of the topmost level in the hierarchy, "New York" is the second highest and so on)

A2: product = Electronics.Computer.HP.Inkjet Pinter
("Electronics" is the value of the topmost level in the hierarchy, "Computer" is the second highest and so on)

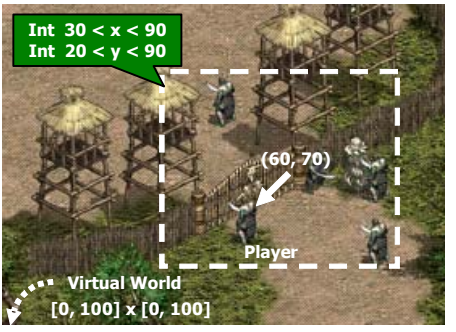


Fig. 3. MMOGs

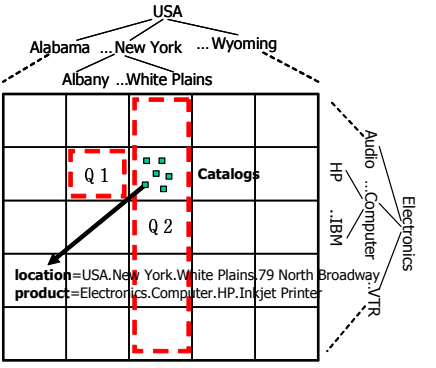


Fig. 4. P2P catalog systems

Dominantly issued queries are two-dimensional range queries such as Q1 and Q2 in Figure 4[1]. Within each level in the hierarchy, partial string matching (*ex.* USA.N*.White Plains) is not usual. Thus, clustering according to the hierarchy is enough, while clustering between similar string values in same level is not necessary.

The hash-concatenation scheme hashes the value of each level in the hierarchy into M / d bits, where d is the hierarchy depth. It then concatenates the hashed values one after another. In practice, d is determined by an application. To hash a variable-length string value of each level to a fixed-length bit representation, a modified SHA-1[20] hash function is used[2]. Queries are also encoded in the same way. The tables below show encoding examples when M is 80 and the hierarchy depth d is 4. Thus, each string in the hierarchy is hashed to 20 bits.

A1: location = USA.New York.White Plains.79 North Broadway ⟶
h_{20}(USA)·h_{20}(New York)·h_{20}(White Plains)·h_{20}(79 North Broadway)

A2: product = Electronics.Computer.HP.Inkjet Pinter ⟶
h_{20}(Electronics)·h_{20}(Computer)·h_{20}(HP)·h_{20}(Inkjet Printer)

•

Q1: location = h_{20}(USA)·h_{20}(New York)·h_{20}(Albany)* ∧
product = h_{20}(Electronics)·h_{20}(Computer)·h_{20}(HP)*

Q2: location = h_{20}(USA)·h_{20}(New York)·h_{20}(White Plains)* ∧
product = * (* means a wild card)

[1] Although `String` type keyword queries such as "%keyword%" are popular in P2P file sharing, we do not tackle such queries because they do not benefit from object clustering.

[2] SHA-1[20] hashes a string to the randomized 160-bit. A string of each level has to be hashed to M / d bits which is less than 160. Thus, just M / d prefix bit of SHA-1 is used as a hashed value.

Our hash-concatenation scheme is useful in two aspects. First, due to hashing, a variable-length string is encoded to a fixed size bit length. Second, locality is preserved due to the concatenation scheme. Bit keys with similar hierarchical structures are closely clustered. In contrast, previous string-to-bit encoding schemes such as serial numbering and prefix encoding [3][19] are not feasible in P2P environments. The serial numbering scheme, which stores all mappings from strings to serial numbers, can not add new values easily. If new values in the hierarchy are added, all peer nodes have to update their mapping. Prefix encoding also can not categorize variable length strings well. Since this scheme encodes only the prefix characters of a string due to limited bit length, objects are not clustered well according to the hierarchy.

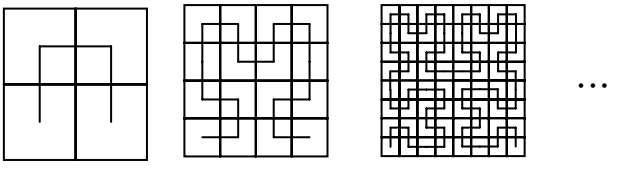


Fig. 5. Hilbert SFC

Step2: Mapping multiple bit keys to a one-dimensional *N*-bit key while considering multi-dimension clustering – Many schemes have been studied to map multi-dimensional keys to a one-dimensional key [2]. Space Filling Curve (SFC) is a well-known scheme. It includes z-ordering, Gray code and the Hilbert SFC. We use the Hilbert SFC because it has better object clustering properties compared to other SFCs. It can be implemented with a simple state machine. As an example, the Hilbert SFC maps {x=1010 ∧ y=1011} to 10001011. The Hilbert SFC has two interesting properties: recursion and locality preservation. Figure 5 shows the recursion. The locality preserving property can be described as follows. Points which are close to each other along the space filling curve map to points which are close in the multi-dimensional space. We utilize this property for the multi-dimensional range query routing protocol.

4.2 Efficient Routing Protocols for Data Updates and Multi-dimensional Range Queries

CISS supports efficient routing protocols to maximize the benefit of object clustering: a *caching-based update routing protocol* for data updates and a *forwarding-based query routing protocol* for multi-dimensional range queries. Both of them significantly reduce the number of costly DHT lookups, and thus improve the efficiency of data-intensive and multi-dimensional range query-intensive P2P applications. We describe the technical details as follows.

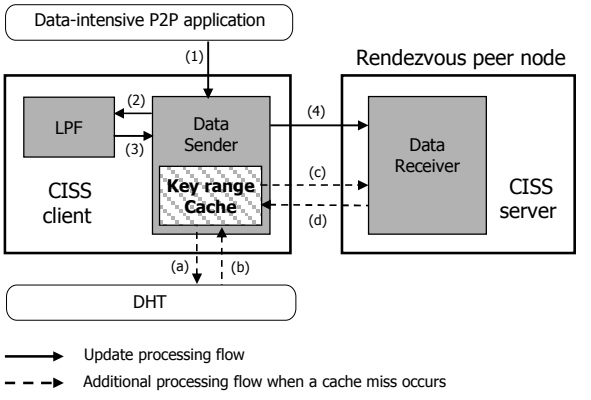


Fig. 6. Caching-based update routing protocol

Caching-based update routing protocol: As shown in Figure 6, the *CISS client* caches the key range of the most-recently-searched rendezvous node. Thus, the CISS client does not perform additional DHT lookups if streams of updates belong to the cached key range (cache hit). Streams of updates belong to the same node with high probability. It is because each peer node manages a semantically contiguous key range and data updates are usually semantically close. For example, in MMOGs, a subsection of the virtual world is managed by a peer node. Players will spend significant amounts of time in a given subsection and therefore their data will belong to the same node with high probability. To quantify the performance benefit of this update routing protocol, we measure the *hit ratio* of the key range cache. In our experiments, we also measure the hit ratio under various data mobility values because the hit ratio is directly affected by a data mobility value. The cached key range can be stale due to DHT topology changes (*e.g.* leave and join of nodes). Thus, a TTL (Time-To-Live) mechanism is utilized to maintain the consistency of the cached key range. After the TTL expires, the CISS client performs a DHT lookup to refresh the cached key range.

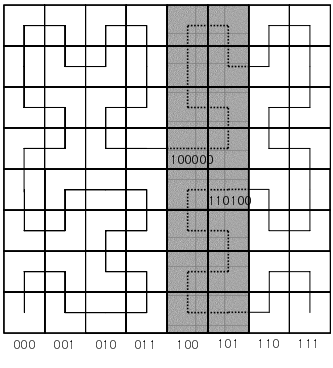


Fig. 7. Forwarding-based query routing protocol

Forwarding-based query routing protocol: For multi-dimensional range query routing, we propose a forwarding-based query routing protocol that reduces costly DHT lookups by forwarding a query to succeeding peer nodes. Multi-dimensional range queries involve multiple contiguous key ranges in CISS. To reduce the number of DHT lookups for those multiple key ranges, the forwarding-based query routing protocol utilizes the object clustering property of CISS. Assume that a user issues a multi-dimensional range query (10*, *) which is mapped to the two dotted curves in the gray area as shown in Figure 7. The Query Requester in CISS client module finds the first keys from the each contiguous curve using LPF, which are 100000 and 110100. The Query Requester then searches matching peer nodes via DHT lookups for the two keys. If matching peer nodes are found, the Query Requester sends the query (10*, *) to the Query Respondents in the CISS server modules of those peer nodes. The Query Respondent generates a result and sends it to the Query Requester. If the key range for the result is larger than the key range managed by the node, the Query Respondent forwards the query to the succeeding peer node without having to perform any more DHT lookups. Query forwarding is repeated until all relevant data are found for the result. In CISS, the number of DHT lookups is determined by the number of separate curves describing the query. In addition, the number of query forwarding messages depends on the size of the query range as well as the topology of a P2P overlay network. In experiments, we show that the forwarding-based routing protocol outperforms existing DHT-based query routing protocols in terms of the number of messages needed for query processing.

In Squid [19], authors suggested a mechanism to resolve a multi-dimensional keyword and range query by embedding a tree structure into the P2P overlay network topology. In this mechanism, all queries should be routed to the peer matching the cluster prefix 0 or 1 for query refinement. Thus, the peer can be the congestion point, which can result in one point of failure. However, our forwarding-based query routing protocol does not incur such a query congestion problem while supporting efficient query processing with few DHT lookups.

4.3 Cluster-Preserving Load Balancing

CISS supports two load balancing schemes: *local-handover* and *global-handover*. In order to achieve load balancing, both of them hand over the partial key range managed by an overloaded node to lightly loaded nodes. However, in contrast to the previous virtual server approach, our approach does not destroy the object clustering property. Thus, it still maintains the benefit of object clustering to process data update and multi-dimensional range queries.

In local-handover, the overloaded node hands over a part of its own key range to one of its neighbor nodes (predecessor or successor). This can be done easily by a leave followed by a join. Figure 8 shows a local-handover example. When node ***B*** gets overloaded, it hands over a part of its key range to its predecessor node ***A*** or successor node ***C***. If ***A*** takes the load of ***B***, ***A*** leaves the DHT-based overlay network and joins again closer to ***B*** so as to adopt the part of B's key range. This reduces the key range which ***B*** must manage, and the ***B***'s key range is therefore decreased.

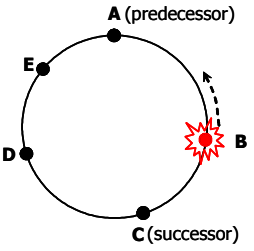


Fig. 8. Local-handover

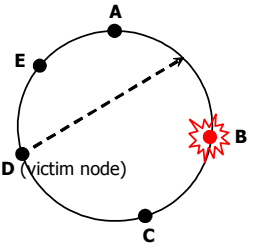


Fig. 9. Global-handover

Table 2. Load Balancing Cost

	Local-handover	Global-handover
DHT routing table updates	• $O(\log S)$ messages	• $O(\log S)$ messages
Object Transferring	• From the overloaded node to the neighbor node	• From the overloaded node to the victim node • From the victim node to the successor of victim node.
Victim Probing	• None	• ***n*** DHT lookups

Similarly, ***C*** can take ***B***'s load if ***B*** leaves and joins again. Even after the local-handover is performed, each node still manages a contiguous key range. Thus, object clustering is preserved. However, cascading load propagation can occur in this scheme. If a neighbor node also gets overloaded due to the local-handover, it will also perform a local-handover to its neighbor node, and so on.

To alleviate this shortcoming, we propose global-handover. In this scheme, an overloaded node hands over a part of its key range to a victim node instead of a neighbor node. After probing randomly selected nodes in the DHT-based overlay network, the most lightly loaded node is determined as a victim node. Figure 9 shows a global-handover example. If node ***D*** is determined as a victim node, an overloaded node ***B*** makes ***D*** leave. Node ***D*** then joins as a predecessor of ***B*** and takes over a contiguous sub-range of ***B***'s key range. Also, ***D***'s successor node ***E*** manages contiguous key range. Thus, object clustering is still preserved.

Table 2 shows the load balancing cost of both schemes. First of all, the cost for updating the DHT routing table is the same since the node leave and the node join occurs only once in both schemes. The cost of the object transferring from the overloaded node to the lightly loaded node is also the same. However, global-handover requires additional object transferring cost as well as a victim-probing cost. The additional object transferring cost is required because the victim node should hand over all of its objects to its successor node before leaving the overlay network. For victim probing, it is necessary to collect load information from n randomly selected nodes. Thus, it requires ***n*** DHT lookups. To minimize the load balancing cost, CISS performs global-handover only when the cascading load propagation is expected if local-handover were to be used.

We are currently investigating some technical details for the proposed schemes including overload detection, load estimation and victim selection algorithms. For efficient load estimation, we are developing a histogram-based algorithm. To hand over the proper amount of load, all nodes have to know their own load information in detail. However, it is not practically possible to maintain load information for each key. Thus, each node divides its key range into several sub-ranges and then maintains a histogram for the number of requests in each sub-range.

5 Experiments

In this section, we demonstrate the performance benefit of CISS compared to existing DHT-based P2P systems which use a hash function. For our experiment, we have implemented a C++-based simulation engine which includes the Hilbert SFC-based LPF, the core functions of the CISS client and server module and a Chord-based DHT overlay network. The simulation has been performed for three overlay network topologies which consist of 1000, 10000 and 100000 peer nodes respectively. The identifier of each node is randomly generated. To exclude the effects of dynamic topology changes, we did not simulate node leaves or joins. We detail the performance of the proposed routing protocol in the simulation results below.

5.1 Data Update Performance

In each simulation, 1000, 10000 and 100000 mobile clients in a virtual world generate their position updates periodically for the workload of the simulator. Before updating its position, the mobile client checks whether its current position is in the cached key range. If a cache miss occurs, it looks up the node that is responsible for its current position. The mobile clients are designed to wander the $[0, 2^{12}] \times [0, 2^{12}]$ square virtual world based on the ns-2 *random waypoint mobility model* [6]. Each mobile client updates its position every 125 milliseconds (for comparison, the first-person shooter Quake II updates a player's position every 50ms); a position consists of two attributes: an x-coordinate and a y-coordinate. The simulation is run for 300 seconds.

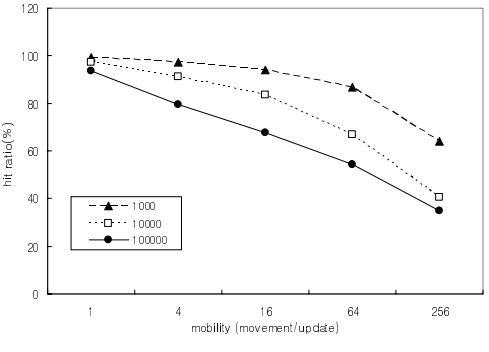


Fig. 10. Hit ratio of the key range cache

To measure the performance benefit of the caching-based update routing protocol, we use the hit ratio of the key range cache. Figure 10 depicts the average hit ratio of the key range cache over all the mobile clients having the same mobility value. A mobility value of 1 means that a mobile client can move maximum one pixel length in the $[0, 2^{12}] \times [0, 2^{12}]$ virtual world during one position update period. From the Figure 10, we see that this update routing protocol significantly reduces the number of look-ups for position updates (by up to 93% with 100000 nodes), whereas with hash function, the mobile client has to look up the node responsible for its current position at every position update. Because the range of mobile client movement is much smaller than the range managed by the responsible server, the hit ratio is high for low mobility values. The larger the mobility value, the lower the hit ratio. However, even with a high mobility value of 256, the update routing protocol achieves a 35% hit ratio with 100000 nodes. Figure 10 also shows the hit ratio variation according to the number of nodes. The range managed by each node increases as the number of peer nodes decreases. Thus, the hit ratio with a 1000 node topology, which has larger range size than the other ones, is the highest.

5.2 Multi-dimensional Range Query Performance

To implement multi-dimensional range queries, we used a P2P catalog system as an example application. The catalog is categorized by two attributes (**location** and **product**); each attribute consists of four levels. We have performed experiments for each of ten query types. For example,

- **Q(4,4):** Queries with both attributes having values in the top four levels of the hierarchy, *e.g.* (**location:** USA.New York.White Plains.79 North Broadway, **product:** Electronics. Computer.HP.Inkjet Pinter).
- **Q(4,3):** Queries with one attribute having values in the top four levels and the other attribute having values in the top three levels. *e.g.*(**location:** USA.New York.White Plains.79 North Broadway, **product:** Electronics.Computer.HP.*).

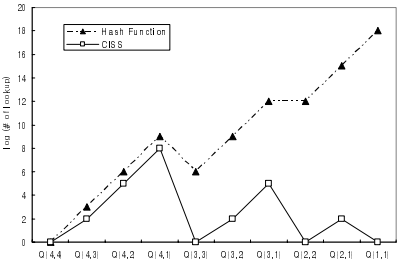


Fig. 11. # of DHT lookups

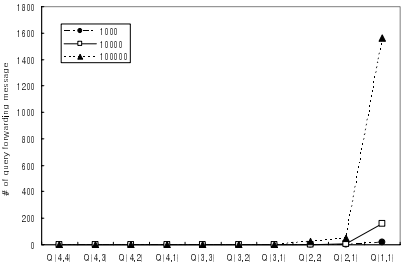


Fig. 12. # of DHT forwarding messages

The other queries Q(4,2), Q(4,1), Q(3,3), Q(3,2), Q(3,1), Q(2,2), Q(2,1) and Q(1,1) are similarly generated. In our experiment, the LPF constructs 24-bit keys. Thus, each

attribute is encoded using 12 bits. We simulated all possible combinations for each of the ten queries.

Figure 11 shows the average number of DHT lookups for the ten types of queries in log-scale. When the size of the query range becomes large, the number of DHT lookups is significantly reduced. This is clearly shown from Q(3,3) to Q(1,1) where the difference in the number of lookups required for CISS compared to the hash-based approach is dramatically illustrated. This is achieved due to the object clustering effect of CISS. Queries like Q(3,3), Q(2,2) and Q(1,1) are mapped to one contiguous curve on the Hilbert SFC. For such queries, only one DHT lookup is necessary for query processing in our forwarding-based query routing protocol. However, Q(4,4) is an exact matching query. Thus, in this case the number of DHT lookups required for CISS is the same as that for the hash-based approach. Finally, in cases Q(4,3), Q(4,2) and Q(4,1), one attribute is specified exactly. These results in a decrease in object clustering and therefore decreased performance benefit. Nevertheless, CISS still performs two times better than the hash-based approach for these queries.

Figure 12 shows the average number of query forwarding messages when all peer nodes manage the same size key range. As shown in the figure, the first nine types of queries with 1000 nodes and seven types of queries with 10000 and 100000 nodes do not need query forwarding. The results for these queries can be retrieved from the peer node found out by a DHT lookup. On the other hand, in the cases of Q(1,1) with 1000 nodes and Q(2,2), Q(2,1), Q(1,1) with 10000 and 100000 nodes, query forwarding is necessary because the query range size is larger than the key range size which the peer node manages. However, the forwarding cost is just one message whereas a DHT lookup may cost several messages. Figure 11 and Figure 12 demonstrate that the total number of messages for query processing is significantly reduced in our forwarding-based query routing protocol.

6 Conclusion and Future Work

We have described CISS, *a framework that supports efficient object clustering for DHT-based peer-to-peer applications*, especially data-intensive and multi-dimensional range query-intensive P2P applications. While utilizing a DHT-based overlay network as a scalable and robust lookup layer, CISS uses a Locality Preserving Function (LPF) instead of a hash function. Thus, CISS achieves a high level of object clustering without requiring any changes to existing DHT implementations. Our simulation studies show that a *caching-based update routing protocol* reduces the number of DHT lookups for data updates by up to 93% with 100000 peer nodes, and a *forwarding-based query routing protocol* for multi-dimensional range queries outperforms existing DHT-based P2P systems by up to an order of magnitude. We are currently developing the cluster-preserving load balancing mechanism in detail.

References

[1] A. Andrzejak and Z. Xu, "Scalable, Efficient Range Queries for Grid Information Services", In *Proceedings of IEEE P2P*, Sweden, September 2002

[2] T.Asano, D.Ranjan, T.Roose,E. Welzl and P.Widmaier, "Space Filling Curves and Their Use in Geometric Data Structures", *Theoretical Computing Science*, 181, 1997, pp.3-15

[3] A.R. Bharambe, S. Rao and S. Seshan, "Mercury: A Scalable Publish-Subscribe System for Internet Games", In *Proceedings of NetGames*, Germany, April 2002

[4] J. Byers, J. Considine and M. Mitzenmacher, "Simple Load Balancing for Distributed Hash Tables", In *Proceedings of IPTPS*, CA, USA, February 2003

[5] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris and I. Stoica, "Wide-area cooperative storage with CFS", In *Proceedings of SOSP*, Canada, October 2001

[6] K. Fall and K. Varadhan. NS Manual

[7] A. Gupta, D. Agrawal and A. El Abbadi, "Approximate Range Selection Queries in Peer-to-Peer Systems",In *Proceedings of CIDR*, CA, USA, January 2003

[8] M. Harren, J.M. Hellerstein, R. Huebsch, B.T. Loo, S. Shenker and I. Stoica, "Complex Queries in DHT-based Peer-to-Peer Networks", In *Proceedings of IPTPS*, MA, USA, March 2002

[9] R. Huebsch, J.M. Hellerstein, N. Lanham, B.T. Loo, S. Shenker and I. Stoica, "Querying the Internet with PIER", In *Proceedings of VLDB*, Berlin, September 2003

[10] B. Knutsson, H. Lu, W. Xu and B. Hopkins, "Peer-to-Peer Support for Massively Multiplayer Games", In *Proceedings of INFOCOM*, Hong Kong, China, March 2004

[11] A. Kothari, D. Agrawal, A. Gupta and Subhash Suri, "Range Addressable Network: A P2P Cache Architecture for Data Ranges", In *Proceedings of IEEE P2P*, Sweden, September 2003

[12] A. Misra, P. Castro and J. Lee, "CLASH: A Protocol for Internet-Scale Utility-Oriented Distributed Computing", In *Proceedings of ICDCS*, Japan, March 2004

[13] V. Papadimos, D. Maier and K. Tufte, "Distributed Query Processing and Catalogs for Peer-to-Peer Systems", In *Proceedings of CIDR*, CA, USA, January 2003

[14] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp and I. Stoica, "Load Balancing in Structured P2P Systems", In *Proceedings of IPTPS*, CA, USA, February 2003

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A Scalable Content-Addressable Network", In *Proceedings of SIGCOMM*, CA, USA, August 2001

[16] S. Ratnasamy, J. M. Hellerstein and S. Shenker, "Range Queries over DHTs", *IRB-TR-03-009*, June 2003

[17] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", In *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Germany, November 2001

[18] O. Sahin, A. Gupta, D. Agrawal and A. El Abbadi , "A Peer-to-peer Framework for Caching Range Queries", In *Proceedings of ICDE*, MA, USA, March 2004

[19] C. Schmidt and M. Parashar, "Flexible Information Discovery in Decentralized Distributed Systems", In *Proceedings of HPDC*, WA, USA June 2003

[20] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", In *Proceedings of SIGCOMM*, CA, USA, August 2001

[21] B. Y. Zhao, J. Kubiatowicz and A. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing", *UCB Tech. Report UCB/CSD-01-1141*