2012 International Conference on Information and Computer Networks (ICICN 2012)
IPCSIT vol. 27 (2012) © (2012) IACSIT Press, Singapore

A Construction Technique of Constant Degree P2P Systems towards Efficient Complex Queries

Wang Xiaohai [1+], Yang Xiaoxiao [2,] Dai Xiuliang[3]

[1, 3] Campaign Command Department, Navy Submarine Academy, Qingdao, China,
[2] Postgraduate Department, Navy Submarine Academy, Qingdao, China

Abstract. Constant degree P2P system is turning into P2P domain's promising hotspot, but most research focus on DHT construction and maintenance, while leaving optimization and supporting to complex query behind. Underlying topology affects upper-layers' character a lot. Constant degree topologies' inherent property make a constant degree P2P system built using classical technique be poor in the data locality, thus unfit for efficient, low-cost complex queries. Aiming at this shortage, a general-purpose construction technique towards efficient complex queries is proposed, which adds an embedding transformation layer between data layer and DHT overlay. In this way, adjacent data are stored in overlay's adjacent nodes and the data locality is improved, so that the number of nodes referred in complex queries can be minimized with a limited time overhead..

Keywords: data locality; constant degree topology; P2P; DHT; complex query.

1. Introduction

A constant degree P2P system is a kind of structured P2P system with O(log*N*) routing efficiency and a constant number of neighbours for each node. Although belonging to the third generation P2P systems, the constant degree P2P enjoys superiority not only in high routing efficiency, but also in less updating, controlling messages, which decreases network load and improves system performance[1]. Thus, many constant degree P2P systems are proposed based on constant degree graph, such as Kautz[2~5], de Bruijn[6], etc., and turn into the hotspot of the P2P domain in recent years.

Data query is a key problem and the most widely usage of P2P systems. More and more applications require structured P2P systems to support a variety of complex queries[7, 8], especially range query. However, it is usually more difficult to build a constant degree P2P system owing to the special rule of constant topology nodes' naming, connection, and routing, which resulting to most constant degree P2P research are still focus on DHT construction and maintenance, while leaving complex query research behind the classic structured P2Ps. Even existing technologies also follow the classic structured P2Ps, which means storage the continuous data objects to nodes with adjacent ID, and define a general routing algorithm to access them. However, these nodes are not adjacent in the topology. After analysing, we found this characteristic of constant degree topologies are quite different to the ring, mesh, tree and other classic topologies, and a constant degree P2P system' locality will be poor if constructed with "classical techniques".

Aiming to improve constant degree P2P systems' complex queries, we present a general-purpose construction technique. It optimizes complex queries' effectively by enhancing data locality without changing underlying overlay. The main idea is mapping data to nodes combing with topology structure. In this way, data locality is enhanced, and distances between similar semantic objects are reduced. As a result, the number of referred nodes in complex queries decreases while ensuring a bounded time overhead.

[+] Corresponding author. Tel.: +86-532-51858525 ; fax: +86-532-51858525
E-mail address: seeseahai@163.com.

2. Locality analysis of constant degree topologies

In this section, we analysis Kautz and the conclusion can be extended to other constant degree topologies.

2.1. Kautz digraph

Given an positive integer d and $n \geq 1$, Kautz digraph K(d,n) is a directed graph with diameter n and all nodes' in-degree and out-degree equal i, each node's ID is a d-based Kautz String $x=x_1x_2\ldots x_n$, where $x_i \in \{0,1,...,d\}, (1 \leq i \leq n)$, $x_i \neq x_{i+1}$ ($1 \leq i \leq n-1$, and links to nodes $x_2\ldots x_n y$, where $y \in \{0,1, ...,d\}$, $y \neq x_n$.

For any two nodes x (x_1x_2 ... x_n) and y (y_1y_2 ... y_n), we can define the complete order relation of the node set: $x \prec y \Leftrightarrow \exists j \in [1,n]\left(\forall i \in [1,j)\right)\left((x_i = y_i) \wedge (x_j < y_j)\right)$

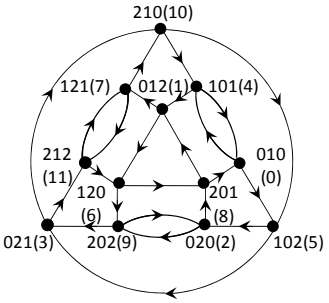


Fig. 1: Nodes and their sequence numbers in K(2,3)

Transformation between ID and nodes' sequence number can also be defined based on this order, Fig1 shows K(2,3) and nodes: because $000 \prec 001 \prec 010 \prec \cdots \prec 110 \prec 111$ in K (2,3), their sequence numbers are 0,1, ..., 11.

K(d,n) uses an bit-shift routing schema, including *long path routing* ρ_l, *shortest path routing* ρ_s, and *undirected routing* ρ_u[9]. ρ_l is the most general schema by n or n-1 times gradually right shifting. ρ_s finds the shortest right-shifting path by founding the longest common sub-string of the source node ID's suffix and destination node ID's prefix; ρ_u is based on undirected digraph and will not limited to right direction, its distance is usually shorter than ρ_l and ρ_s.

2.2. The locality difference

Data locality is an important condition to support range query. Because other complex queries[7, 8] can be considered as adding specific semantic operations on range query, they also need data locality to reduce query overhead. However, P2P systems have different data localities at different logical levels. From the point of structured P2P's building process[10], we can define:

ID locality: data locality in ID space, which means data objects' ID have a smaller absolute difference.

Routing locality: data locality in DHT overlay, which means data objects are stored in nearby nodes.

It can be seen that the locality required in complex queries should be *routing locality*, because it limits the accessing and interaction within a local area of overlay and then ensure a low-cost query indeed. In classic DHTs' topologies, such as ring, grid, and tree, etc, routing between nodes with adjacent ID can be completed in one or a few hops. In another word, *ID locality* and *routing locality* is consistent to a certain extent: adjacent ID means smaller distance in topology, thus, the classical constructing method can ensure the data locality of classic DHTs.

However, the constant degree topologies are different: nodes with adjacent IDs(or sequence numbers) do not always lives nearby: both Butterfly and CCC's routing distance are determined by ID's XOR value. For example, 01111 and 10000 have a numerical difference of 1, while a routing distance of 5. The bit-shift routing schema also makes Kautz and de Bruijn's routing distances are decided by shifting distance instead of sequence numbers. This characteristic determines that, classic construction technique can't guarantee the constant degree P2P system's *routing locality*: although data objects with similar semantic are stored by nodes with adjacent IDs, these nodes are still distributed dispersedly. Thus, more non-relevant nodes are needed to forward message during complex queries, resulting to higher overhead and node occupancy rate.

3. The construction technique towards efficient complex queries

3.1. Model and definitions

We define space I_P and I_R, which contain complete resource set R and node set P, and scale are as same size as ID space I. Then the distribution of resources can be seen a map between I_P and I_R. The mapping in the classic construction technique is an equivalent map between I_P and I_R. If we improving the mapping function, nodes and data objects can be combined more flexible, and some data distributed property needed is also can be achieved, such as routing locality.

Data Topology: the topology that contains I_R, with its structure also reflects locality, denoted as *F*.

Node set V(*F*) is a partition of I_R. Node is the minimum mapping unit, and edge set E(*F*) is the connection between adjacent partition subsets. Therefore, F has a one or multi-dimensional mesh structure, and data with adjacent semantic must be located in one node or neighbours.

Embedding of data Topology: a map form *F* to node Topology, ϕ:V(*F*)→V(G), so any edge (*x*, *y*) in *F* has a path ($\phi(x)$, $\phi(y)$) in *G* as its image.

The longest path length is the expansion number *dil*(ϕ). It is easy to see, a smaller *dil*(ϕ) is always means the better *routing locality*, and *dil*(ϕ)=1 means ϕ is isomorphic.

By adding an embedding transformation layer as an optimization layer, we propose the construction technique towards efficient complex queries:

Step 1: Define the ID space I, and select function that mapping recourse set R and node set P to I;

Step 2: Determine the dimension and partition of R, and construct the data topology *F*;

Step 3: Select *F*'s embedding function ϕ: V(*F*)→V(*G*), and complete the partition of R. Then each node's sub-space in the topology is determined, for example, node *p* will be responsible for the union of all data topology nodes' original image in R, i.e., $subspace(p)=\cup\{x:x\in F,\phi(x)=p\}$;

Step 4: Connect nodes to form the DHT overlay, define routing and topology maintenance strategy.

Step 5: Define algorithm of data joining and query, add locality maintenance strategy.

By abstracting and embedding the data topology, the technology merges topology information organically, and avoids constant degree topologies' defect effectively. It can be used to improve a variety of "pure constant degree" DHTs, such as FissionE, Moore, ORDI and so on.

3.2. DHT reconstruction algorithm

We reconstruct FissionE in this section, including matching strategy between data and nodes, re-allocation of resources under the dynamic environment, query strategy and the locality maintenance strategy.

- **Matching strategy**

Base diameter: diameter of the Kautz graph that FissionE topology is mostly closed to, denoted as $\tilde{n}$. A node with ID length equals $\tilde{n}$ is a *base node*, and its sub-space is a *base space*.

There are about 80% nodes are *base node* in FissionE because of its balance characteristics[5]. Thus, we can determine $\tilde{n}$ easily by adding statistical information into heart beat message. In addition, the embedding function should be distributed and without global information, so we define the matching strategy based on the recursive construction algorithm of undirected Kautz Hamilton path, which complexity is $O(n^2)$[11].

$K(d,\tilde{n})$ has $N=d^{\tilde{n}}+d^{\tilde{n}-1}$ nodes, assuming the one-dimensional data space is [*L*,*H*], then the area of base subspace will be (*H*-*L*)/*N*. for a node *x*, if $|x|=\tilde{n}$, then *x* is a base node, and its sequence number $s=path\_sn(x)$, its responsible sub-space *subSpace*(*x*) will be:

$$\begin{cases} [L, L+(H-L)/N] & s=1 \\ (L+(s-1)(H-L)/N, L+s(H-L)/N] & 1<s\le N \end{cases}$$

For non-base node, if $|x|<\tilde{n}$, then *subSpace* (*x*) is the union of all base nodes who has prefix *x* :

$$subSpace(x)=\cup\{subSpace(q):q\in K(2,n), q.startwith(x)\}$$

If $|x|>\tilde{n}$, suppose *x'* is the prefix of *x* with length $\tilde{n}$. if *subSpace*(*x'*)=(*l*,*h*]. Then *subSpace*(*x*) can be calculated by dividing (*l, h*]. Let $\Delta n=|x|-\tilde{n}$, then *x'* could be divided into $2^{\Delta n}$ nodes with length |*x*|, note *x*'s sequence number is *k*, then $subSpace(x)=(l+(s-1)(h-l)/2^{\Delta n}, l+k(h-l)/2^{\Delta n}]$, $1\le k\le 2^{\Delta n}$。

Data object *o* should calculate which base space it belongs to firstly, and then find the base node *p* that responsible for it using *Kautz-id*. After that, it can route to *p* or another non-base node *p'* using FissionE routing algorithm. *p* will storage *o* if $|p'|<\tilde{n}$. Otherwise, *o* will be storage by an node *p"* with prefix *p*, and we can further route to *p"* according to matching strategy. Table 1 shows the algorithm.

- **re-allocation of resources when nodes join or depart**

Supposing node *q* joins the system and will split *subSpace*(*p*). If $|p|=\tilde{n}$, *subSpace*(*p*) is a continuous interval (*l*,*h*], and *p q* will be responsible for (*l*,(*l*+*h*)/2] and ((*l*+*h*)/2,*h*] respectively. To ensure locality, the connection between them should be added on the basis of routing table update algorithm, and the broken connection between node and its successor on the Hamilton path should also be reconstructed.

If $|p|>\tilde{n}$, *subSpace*(*p*) is a continuous interval (*l*,*h*] as well, and its split method is also same to the above case, only with an additional connection(*p*,*q*);

If $|p|<\tilde{n}$, *subSpace*(*p*) contains several un-continuous intervals. After divided, *p* and *q* become sibling nodes with prefix *p*, and no additional routing table updating is needed.

TABLE I. DATA OBJECT JOIN ALGORITHM

Algorithm *data_object_join*(*data*)

Determine the standard range *SR* which *data* belongs to
standard node *p* = *Kautz-id*(*SR*)
FissionE_routing_to(*p*)
If($|p|\le\tilde{n}$) **then**
store(*data*, *p*)
If($|p|>\tilde{n}$) **then**
select sibling node *p"*that manages sub-range of *data*
store(*data*, *p"*)

TABLE II. RANGE QUERY ALGORITHM

Algorithm *flood_range_query*(rangeQuery *r*)
Caculate *r.range* 's the middle point *p* Send *r* to *p* *flood_routing* (*p, Φ, Query*)
Procedure *flood_routing* (*pt, sender, Query*)
foreach(neighbour *nb* of *pt*) If((*subSpace*(*nb*)∩*r.range*≠Φ),*nb*!= *sender*) Send *Query* to *nb* *flood_routing* (*nb, pt, Query*)

- **Query Algorithm**

The Keyword and range query algorithms should adapt to the new matching strategy. Keyword query algorithm is as same as the node joining algorithm, while range query needs to reach a node in the range first and then gradually spread to whole range. We use a "flooding on the path" approach to develop range query algorithm: a node that receives a query message will forward it to all neighbours among the range.

For one-dimensional data, range query can reduce the number of referred nodes while ensuring a convergent, bounded time overhead. Concretely, we have:

Theorem 1: when flooding among *M* nodes in K(*d*,*n*)'s Hamilton path, the flooding time *Flood_T* and the number of message *Flood_Cm* fulfills: $\log_{2d-1}M < Flood\_T \le M/2$, $M-1 \le Flood\_Cm < d*M$.

Proof: Supposing *p* is the midpoint of *M* nodes. If flood path doesn't include non-path connection, *Flood_T* will be maximum *M*/2, and *Flood_Cm* will be minimum *M*-1. If all neighbors are belonging to these *M* nodes, then the (2*d*-1)-ary tree contained in the path has a lower height bound $\log_{2d-1}M$, while *Flood_Cm* reaches its upper bound. Because nodes will not transfer message to its upstream nodes, and there are 2*d***M* connections in all, thus the *Flood_Cm* has upper bound *d***M*. □

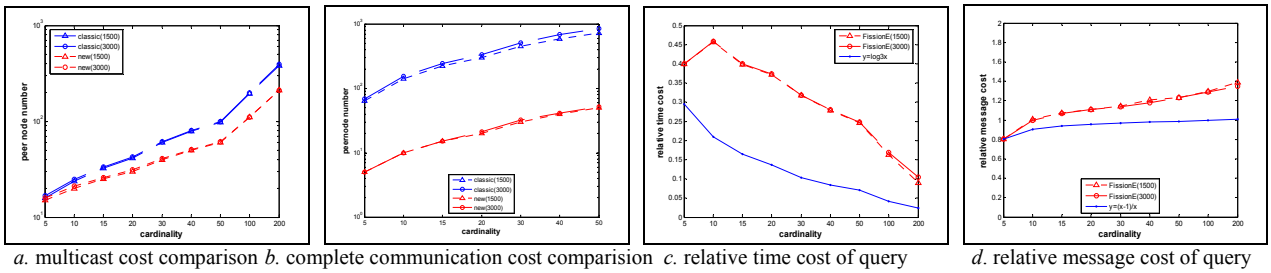
4. Experiments

Firstly, we implement FissionE with scale 1500 (ñ = 10), 3000 (ñ = 11) on peerSim and verify the new technique's performance on the number of referred nodes during query. Fig 2.a compares the number of referred nodes as multicast cost, *classic* and *new* curve present these two construction techniques. The classic constructed overlay transfer message using the PIRA-like method[5] based on routing tree, while the new constructed overlay uses *flood_range_query*. Fig 2.b compares the complete communication cost, the number of referred nodes when every node interaction with each other, just like the Divide and Conquer Skyline[8] and top-*k* query.

The number of referred nodes reflects the node resource's utilization rate. A smaller number means higher rate. Experiments show the new technology decreases the number of referred nodes. If there are interactions between target nodes, this number decreases greatly.

To measure time overhead and verify the convergence of *flood_range_query*, we calculate the relative flooding messages number *Flood_Cm'* and time *Flood_T'*, where *Flood_Cm'*=*Flood_Cm/n*, *Flood_T'*=

Flood_T'/n. Fig 2.c and 3.d show *Flood_T'* and *Flood_Cm'* under different overlay scale and lower bound $y=\log_3 x/x$ and $y=(x-1)/x$. We can find that overlay scale affects *Flood_T'* a little and has a decrease trend when *n* increases; *Flood_Cm'* is only about 30% higher than the lower bound, and get closer to it and farther away from the upper bound while scale increases. Experiments prove that *flood_range_query* reduced the message propagation time (*Flood_T'*<0.5) greatly with a small message overhead. The rapid efficiency's increase and overhead's convergence demonstrated the idea of flooding fits the complete, one to many, and many to many interactions in the constant degree topologies.



a. multicast cost comparison *b*. complete communication cost comparision *c*. relative time cost of query *d*. relative message cost of query

Fig. 2 Experiments

5. Conclusion

Without changing underlying overlay, the general construction technique adds a transformation layer to improve data locality, and optimizes complex queries effectively. In future work, we will study the data distribution pattern further, such as the embedding of high dimensional data, and develop appropriate and efficient routing and query algorithms.

6. References

[1] S. Ratnasamy, S. Shenker, and I. Stoica. Routing Algorithms for DHTs: Some Open Questions[C]// Proc. of IPTPS 2002. Cambridge, USA, 2002: 45-52

[2] D. Li, X. Lu, and J. Wu. FissionE: A scalable constant degree and low congestion dht scheme based on kautz graphs[C]// Proc of IEEE INFOCOM 2005. Miami, Florida, USA, March: IEEE, 2005: 1677-1688

[3] D. Guo, J. Wu, H. Chen, X. Luo. Moore: An Extendable Peer-to-Peer Network Based on Incomplete Kautz Digraph with Constant Degree[C]// Proc of IEEE INFOCOM 2007. Computer Society Press, Los Alamitos

[4] Y. Zhang, X. Lu, D. Li. SKY: Efficient Peer-to-Peer Networks Based on Distributed Kautz Graphs[J]. Science in China Series F, 52(4), April 2009 :588-601.

[5] D. Li. Research on peer-to-peer resource location in large-scale distributed systems [D]. Changsha: National University of Defense Technology, 2005 (in Chinese).

[6] D. Loguinov, A. Kumar, V. Rai, S. Ganesh. Graph-Theoretic Analysis of Structured Peer-to-Peer Systems: Routing Distances and Fault Resilience[C]//Proc of ACM SIGCOMM 2003, Karlsruhe, Germany:395-406

[7] A. Vlachou, C. Doulkeridis, K. Norvag, M. Vazirgiannis. On Efficient Top-k Processing in Highly Distributed Environments[C]// Proc of ACM SIGMOD 2008. Vancouver, BC, Canada 2008:753-764

[8] L. Chen, B. Cui, H. Lu, et al. iSky: Efficient and Progressive Skyline Computing in a Structured P2P Network[C]// Proc of IEEE ICDCS 2008. Beijing, China, 2008: 160-167

[9] J.W.Mao and C.B.Yang, Shortest path routing and fault-tolerant routing on de Bruijn networks[J]. Networks,2000,35(3): 207-215

[10] K. Aberer, L. Alima, et al. The Essence of P2P: A Reference Architecture for Overlay Networks[C]// Proc of Peer-to-Peer Computing 2005. Konstanz, Germany, IEEE,2005: 11-20

[11] Rabah Harbane and Marie-Claude Heydemann. Efficient reconfiguration algorithms of de Bruijn and Kautz networks into linear arrays[J], Theoretical Computer Science,263, 2001:173-18