Improving Routing Load Balance on Chord

Lirong LIN[a], Keiichi KOYANAGI[a]
Takeshi TSUCHIYA[b], Tadashi MIYOSAWA[b], Hiroo HIROSE[b]
[a]Graduate School of Information, Production and System, Waseda University, Japan
[b]Tokyo University of Science, Suwa
linlirong@ruri.waseda.jp, keiichi.koyanagi@waseda.jp
t-tsuchi@rs.suwa.tus.ac.jp, miyosawa@rs.suwa.tus.ac.jp, hirose@re.tus.ac.jp

***Abstract*— Structured P2P overlay networks provide rather balanced query routing load than centralized network because of their distributed design. But certain designing issues might exist and lead to an unbalanced routing load. In some systems like Chord where stored objects are small, routing dominates the cost of publishing and retrieving an object. How to balance the routing load fairly becomes critical. In this paper, we analyse three designing issues that cause an imbalance routing load on Chord and external factor like non-uniform request distribution that aggravates those issues. We aim to evaluate our proposal under highly skewed request distribution and the simulation result shows that our proposal performs great, the routing load fairness among peers are significantly improved, and also has a better query performance after comparing with original Chord and one of the existing enhanced proposal.**

***Keywords*— overlay networks, Chord protocol, load balance**

I. Introduction

Due to distributed design, structured P2P overlay networks provide more balanced data retrieval loads than systems using centralized architecture. In some systems like Chord [1] where stored objects are small, routing dominates the cost of publishing and retrieving an object. Fairly balancing the routing load becomes critical.

Chord is a famous and one of the most representative structured overlay network protocols. In Chord, peer ID and data ID have the same bit length m and are generated by hash function (such as SHA-1 [2]) in the same ID space 2^m. Peer and data form a ring topology in increasing order of ID and data objects are stored in the first peer succeeding their ID. Each peer with peer ID *NID* has a successor list and a routing table called finger table with m entries. The ith entry stores the first peer that succeeds the ID: $(NID + 2^{i-1}) \bmod 2^m$ denoted as $Successor\ (NID + 2^{i-1})$. The finger table and the sketch of fingers are in Figure 1.

Original finger table		
Finger	PeerID	IP Addr:port
1	Succesor(NID + 2^0)	*.*.*.*:*
…	…	…
m-3	Successor(NID + 2^{m-4})	*.*.*.*:*
m-2	Successor(NID + 2^{m-3})	*.*.*.*:*
m-1	Successor(NID + 2^{m-2})	*.*.*.*:*
m	Successor(NID + 2^{m-1})	*.*.*.*:*

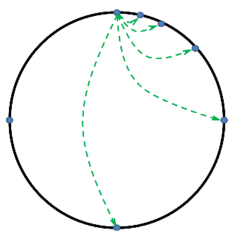


Figure 1. Finger table and fingers sketch

Once a peer searches for a data object, first it checks if its successors succeed the data ID, if not then select the finger peer whose ID is closest to but not larger than the data ID as the next-hop. The query is recursively apply this strategy until reach the target peer.

The rest of the paper is organized as follows. In Section II, related works are introduced. Section III analyse issues that cause the imbalanced routing load in Chord. For each issue, enhanced proposal is proposed and theoretically analysed in Section IV. Section V introduces the metric to evaluate the load balance performance. Simulation results and evaluation are included in Section VI. We conclude our study and point out possible directions for future work in Section VII.

II. Related Works

Many literatures aim to improve the routing load on structured P2P overlay network. In [3], M. Bienkowski et al. utilise methods to manipulate the peer ID generating procedure to ensure peers' interval lengths differ at most by a constant factor to mitigate one of the designing issues. In [4], P. B. Godfrey and I. Stoica use virtual servers instantiated by physical node to act as peers in the network. Once a node becomes heavily loaded, it transfers some of its virtual servers to a proper node with fewer loads. Paper [5] also imports virtual servers. However, above proposals suffer from high maintenance overhead and extra complexity. Paper [6] presents a different way by selecting finger peer dynamically among certain local area. This simple enhancement of finger selection mechanism improves load fairness on Chord substantially.

In our paper, the first proposal allowing queries forward anticlockwise is a new approach to balance routing load on Chord. This "two direction" concept is inspired by paper [7] which introduced 2-Chord. Paper [7] didn't consider load balance and it's differing from finger table and routing strategy with our proposal. We proposed a much more enhanced strategy than that to tackle our load balancing issues. Our third proposal balancing routing load among fingers' local area, is benefited from paper [8], we modified this proposal by enlarging the local area after the author's permission. To evaluate the enhancement, we compared our proposal with original Chord and also with paper [8].

III. Issues Cause Unbalanced Routing Load

In this section we present three designing issues causing unbalanced routing load on Chord and the external factor that aggravate those disadvantages.

A. Single Direction Query Hopping Drawback

For one certain target peer, all the queries from senders are forwarded clockwise to the target peer though the target peer is much closer on the anticlockwise. Consequently, peers beside the right of the target peer bear much more routing loads than peers on the other side in Figure 2.

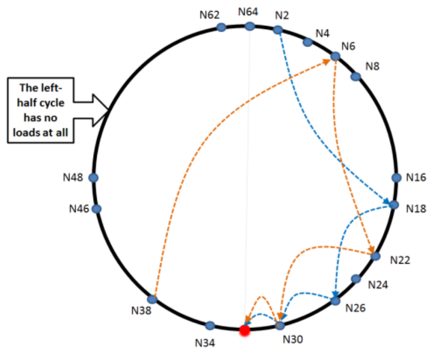


Figure 2. Paths between senders and receiver

B. Greedy Query Hopping Strategy Drawback

During the search period, each query hop reduces the distance to the target greedily. That's to say, queries are gradually forwarded into narrower and narrower ranges (as Figure 2). Intuitively, for a certain target peer, peers that are closer to it will receive much more queries than peers far from it.

C. Various Interval Lengths Between Peers

Peer's ID is generated by hashing the node's IP address. Consequently, the intervals between peers in the ID space may vary considerably. An analysis from paper [9] indicate that the longest interval could be *O(log N)* times longer than the shortest one (N denotes the network size).

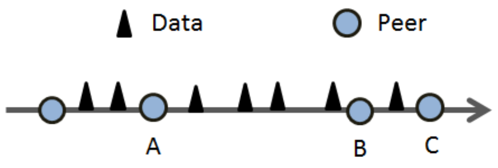


Figure 3. Peer after different interval

Figure 3 is part of a flatted Chord ring; each peer is responsible for the data on the left interval. Peer B is responsible for more data than A and C. According to the algorithm with which the peer selects its routing table entries, the probability of a certain peer being chosen as a routing table entry by distant peers is proportional to the length of its corresponding interval. Thus, peer B is more likely to appear as finger peer of distant peers than A and C. As finger table is used as routing table, finger peer is the routing entry; Queries have more possibility to forward to B and causing an unbalanced routing load distribution among this local area.

However, the first two issues can be cancelled out from a global view if the request distribution is uniform (i.e. all the peers' responsible data has the same popularity). But the external factor is, in real P2P world, the request distribution is non-uniform and some are even highly skewed. Observations in paper [10] show that the actual request distribution in many P2P systems follows the *Zipf's Law*. Peers continuously arrive and depart, or churn phenomenon act as the external factor that aggravate issue C. Peers joining and leaving the network will change the interval length of their neighbours; this may aggravate the difference of interval lengths.

With three designing issues and two external factors that aggravate those issues, a very imbalanced routing load distribution will arise in original Chord.

IV. Proposal Tackle on Each Issue

Three issues causing imbalance routing load have been pointed out in Section III. In this section, proposal is proposed to tackle on each issue.

A. Allowing Query Forward to Anticlockwise

Issue A denote that one direction hopping do has drawback. So proposal A aims to allow query forward anticlockwise. To achieve this, a new finger table point to anticlockwise is added to each peer and each peer contains a successor list and a predecessor list. The routing strategy also has changed.

1) Enhanced Finger Table: Each peer stores two finger tables. One is the clockwise finger table remain as the original one without the last entry, the other one is new added named as anticlockwise finger table.

Clockwise finger table		
Finger	PeerID	IP Addr:port
1	Succesor(NID + 2^0)	*.*.*.*:*
...	...	...
m-3	Successor(NID + 2^{m-4})	*.*.*.*:*
m-2	Successor(NID + 2^{m-3})	*.*.*.*:*
m-1	Successor(NID + 2^{m-2})	*.*.*.*:*
Anticlockwise finger table		
Finger	PeerID	IP Addr:port
1	Succesor(NID - 2^0)	*.*.*.*:*
...	...	...
m-3	Successor(NID - 2^{m-4})	*.*.*.*:*
m-2	Successor(NID - 2^{m-3})	*.*.*.*:*
m-1	Successor(NID - 2^{m-2})	*.*.*.*:*

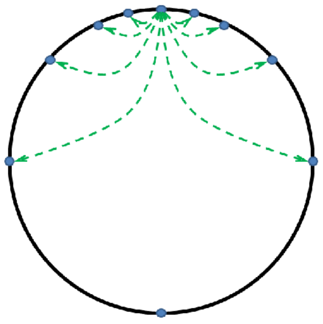


Figure 4. Enhanced finger table and fingers sketch(A)

Figure 4 is the finger tables and the sketch of fingers. Each finger table has *(m-1)* entries, *Successor ()* share the same meaning mentioned in Section I.

2) Enhanced Routing Strategy: Before query forward, peer will check whether the target data ID on its clockwise half ring or on its anticlockwise half ring by comparing its ID with data ID. If target data ID is on the clockwise half ring, clockwise finger table will be chosen as the routing table and the routing strategy remain the same as original one. If not, take the anticlockwise finger table.

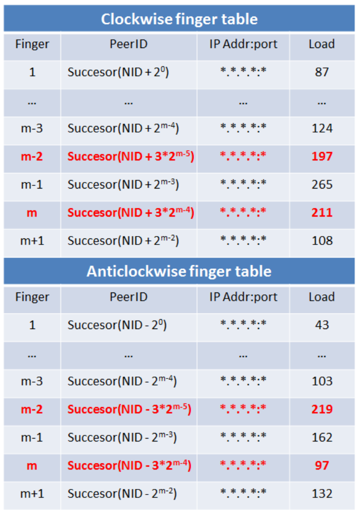
With two available hopping directions, the longest distance between sender and target peer is roughly 2^{m-1} in ID space, rather than 2^m in original Chord. Thus, the max-hop number is one less and the average-hop number is approximately 0.5 less than original. Assume R queries from different senders are forwarded to one certain hot peer. In original Chord, all the queries are forwarded clockwise and the routing loads are gathering on the anticlockwise of the hot peer. After applying this proposal, number r of R queries will be forwarded anticlockwise. Intuitively, part of the loads will divert to the other side of the hot peer. Thus, this proposal can balance the loads among the hot peer' local area, and also improve the overall load fairness in some degree.

B. Lessen Greed of Query Hopping

When peer selects the finger peer nearest to the target key, say k^{th} longest finger, the $(k+1)^{th}$ longest finger is also considered and select the least loaded one as the next-hop. However, under the finger table in proposal A, the distance between 1^{th} and 2^{th} longest finger is approximately 1/8 (1/4 under original one) of the ring. The query will delay 1/8 of the ring at most if take a shorter finger. So two fingers are added to make a better trade-off. One is in the middle of the 1^{th} and 2^{th} longest finger in identifier space. The other one is in the middle of the 2^{th} and 3^{th} longest finger.

To achieve this, each peer stores its own routing load information (number of queries received and subsequently forward to next-hop), and this information is added to the finger table as a new column.

1) Enhanced Finger Table: As shown in Figure 6, each finger table has *(m+1)* entries, the red colour entries are the new added ones. The sketch of fingers is shown in figure 6. The dotted lines are the added fingers.



Clockwise finger table			
Finger	PeerID	IP Addr:port	Load
1	Succesor(NID + 2^0)	*.*.*.*:*	87
...	...	...	...
m-3	Succesor(NID + 2^{m-4})	*.*.*.*:*	124
m-2	**Succesor(NID + 3*2^{m-5})**	***.*.*.*:***	**197**
m-1	Succesor(NID + 2^{m-3})	*.*.*.*:*	265
m	**Succesor(NID + 3*2^{m-4})**	***.*.*.*:***	**211**
m+1	Succesor(NID + 2^{m-2})	*.*.*.*:*	108

Anticlockwise finger table			
Finger	PeerID	IP Addr:port	Load
1	Succesor(NID - 2^0)	*.*.*.*:*	43
...	...	...	...
m-3	Succesor(NID - 2^{m-4})	*.*.*.*:*	103
m-2	**Succesor(NID - 3*2^{m-5})**	***.*.*.*:***	**219**
m-1	Succesor(NID - 2^{m-3})	*.*.*.*:*	162
m	**Succesor(NID - 3*2^{m-4})**	***.*.*.*:***	**97**
m+1	Succesor(NID - 2^{m-2})	*.*.*.*:*	132

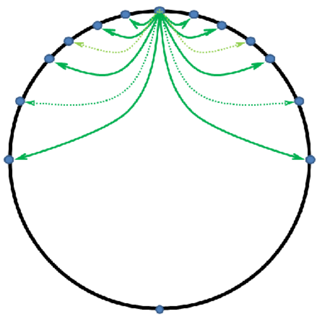


Figure 5. Enhanced finger table and sketch(B)

2) Enhanced Routing Strategy: When the $(m-1)^{th}$, m^{th} or $(m+1)^{th}$ entry is chosen under original greedy strategy, the $(m-2)^{th}$, $(m-1)^{th}$ or m^{th} entry should also considered accordingly. Then chose the least loaded one as next-hop.

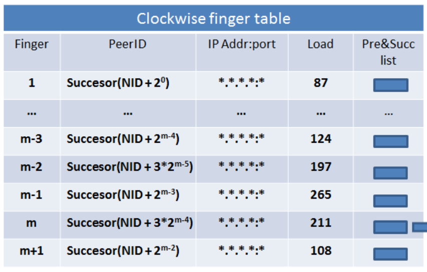
For one certain hot peer, this proposal diverts its routing loads outside (to far peers). So the global load fairness will be improved. While because other hot peer's infection (their loads also be diverted outside), the local load fairness may be sightly influenced. The average hop will increase because of the greed has been lessened.

C. Balancing Routing Loads Among Fingers' Local Area

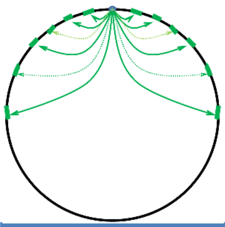
This proposal is benefited from paper [8]. As mitigating interval length differences is difficult and may produce unacceptable overhead. We decide to tackle this issue indirectly. Peer after a larger interval will receive much more routing queries, we can divert its routing load to its neighbours.

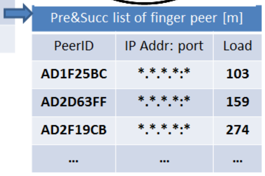
To achieve this, except the information already stored in previous proposals, a list of finger peer's immediate neighbours (with routing load information) is attached.

1) Enhanced Finger Table: Figure 8 presents the enhanced clockwise finger table; the anticlockwise finger table follows the same policy. Now the fingers are no longer point to a peer but a range of peers. Sketch of fingers showed in Figure 6.



Clockwise finger table				
Finger	PeerID	IP Addr:port	Load	Pre&Succ list
1	**Succesor(NID + 2^0)**	***.*.*.*:***	**87**	▬
...	...	...	...	...
m-3	**Succesor(NID + 2^{m-4})**	***.*.*.*:***	**124**	▬
m-2	**Succesor(NID + 3*2^{m-5})**	***.*.*.*:***	**197**	▬
m-1	**Succesor(NID + 2^{m-3})**	***.*.*.*:***	**265**	▬
m	**Succesor(NID + 3*2^{m-4})**	***.*.*.*:***	**211**	▬
m+1	**Succesor(NID + 2^{m-2})**	***.*.*.*:***	**108**	▬





Pre&Succ list of finger peer [m]		
PeerID	IP Addr: port	Load
AD1F25BC	***.*.*.*:***	**103**
AD2D63FF	***.*.*.*:***	**159**
AD2F19CB	***.*.*.*:***	**274**
...	...	...

Figure 6. Enhanced finger table and sketch(C)

2) Enhanced Routing Strategy: Under the strategy of previous proposal, finger peer's neighbours will also be considered and select the least loaded one as next-hop.

With this proposal, although peers may not directly be chosen as finger peer by distance peer. They can still be selected as next-hop and receive forwarded queries. Since sender always selects the least loaded one in finger peer's local area to forward the query, each of such local area in the ring can be automatically balanced. Consequently, local and global load balance will have a qualitative leap. Because finger table holds more information, the average number of query hops will be reduced.

Each proposal improves routing load fairness in different aspect. For one certain hot peer: Proposal A diverts some of its routing load from anticlockwise side to clockwise side, thus the local load fairness will be improved. Proposal B diverts part of its routing load from close neighbours to far peers, so the global load fairness will increase but this may affect other hot peers' local routing load fairness. Proposal C makes a qualitative leap on the local and global routing load balance by averaging routing load of different local areas of the ring. Proposal A, B and C are combined as our final proposal.

V. Routing Load Balance Metric

Jain's Fairness Index [11] is a wildly accepted way of evaluating the fairness among statistical data.

$$JFI = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

We apply the calculation of JFI on the number of routed queries (routing load information) to represent the routing load fairness. The value of JFI can vary from *1/n* to *1*, where 1 indicates the optimal fairness and *1/n* is the worst case. We also dispose the routing load distribution data into statistical figure to make an intuitive view.

VI. Simulation and Evaluation

Our simulation is applied in PeerSim [12], a wildly used P2P network simulator. We focus on skewed and much more skewed request distribution network and carry out a series of simulations in order to verify the effectiveness of those proposals. We also compare our proposal with original Chord and paper [8] denoted as Zhou's proposal.

A. Network setting and Request Distribution

In the following simulations, the network size is set to *10,000* peers (N = *10,000*). PeerID length is *32* bits, successor list and predecessor list size is *16*.

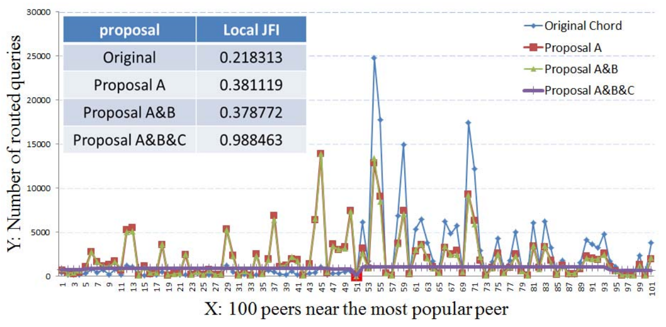
First we form a skewed request distribution network with *500* hot peers and the hottest peer is queried *100,000* times. Then a much more skewed request distribution network with only *50* hot peers and the hottest peer is queried *100,000* times is tested. Queries are initiated at random peers in the overlay network and the number of queries for each popular object is calculated according to the Zipf's Law [10].

Zipf's Law is an empirical law of statistics that roughly show the request distribution on popular objects. When the most popular object is requested *K* times, the i^{th} popular object is requested approximately *K/i* times and so on so forth.

B. Simulation Results

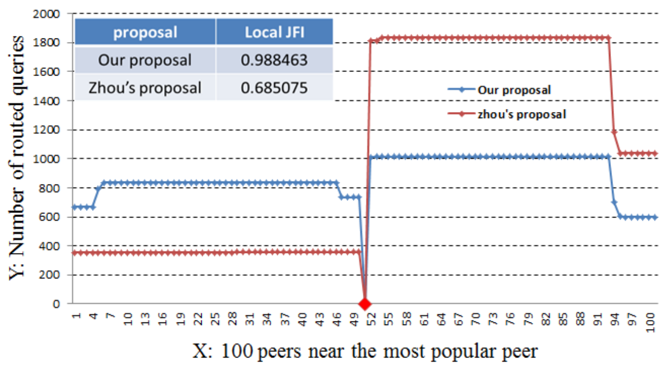
1) Local Load Balance of The Hottest Peer: We observe the routing load distribution of *50* peers beside the clockwise of the most popular peer and *50* on the other side.

Under skewed request distribution (*500* hot peers, 5% of the network):



proposal	Local JFI
Original	0.218313
Proposal A	0.381119
Proposal A&B	0.378772
Proposal A&B&C	0.988463

Figure 7. Load distribution near the most popular peer (500 hot peers)



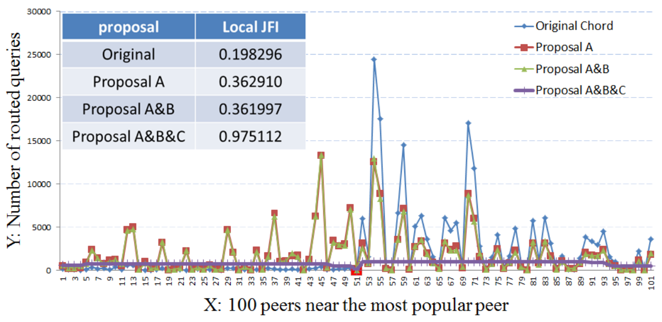
proposal	Local JFI
Our proposal	0.988463
Zhou's proposal	0.685075

Figure 8. Compare with Zhou's proposal (500 hot peers)

From Figure 7 we can see that, in original chord, the left side of the hottest peer bear few loads and the load numbers in the right side is highly irregular form *0* to *25,000*. The local JFI *0.2183* prove that local load distribution in original Chord is extremely unbalanced. After applying proposal A, we can directly see that almost half of the load in the right side has diverted to the left and local JFI *0.3811* confirms proposal A do improve the local load balance in some degree. Proposal A&B and proposal A are almost overlapped but we can see the local JFI *0.3788* has sightly decreased, so we can infer that Proposal A&B may decrease the local routing load balance of the hottest peer sightly because other hot peers' affection just as we analysed in Section IV-B (Actually Proposal A&B is used to improve global load balance). Our final proposal A&B&C make a qualitative leap on the load balance, the local JFI *0.9885* is almost reach *1* and the routing load distribution broken lines almost form a line which means loads have extraordinary balanced.

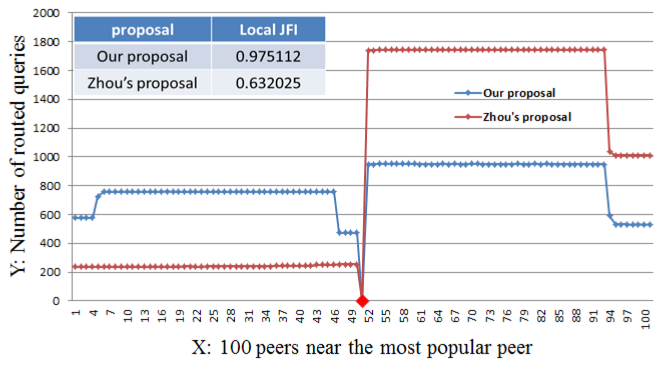
Figure 8 is the local load distribution of our final proposal compare with Zhou's proposal under the same network condition. Intuitively, our proposal is better than Zhou's. The local JFI also prove that our proposal is much better on the local load balance of the hottest peer.

Now let's see the performance under even highly skewed request distribution network (*50* hot peers, 0.5% of the network):



proposal	Local JFI
Original	0.198296
Proposal A	0.362910
Proposal A&B	0.361997
Proposal A&B&C	0.975112

Figure 9. Load distribution near the most popular peer (50 hot peers)



proposal	Local JFI
Our proposal	0.975112
Zhou's proposal	0.632025

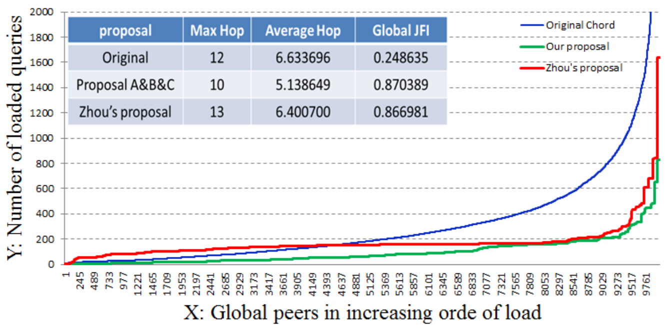
Figure 10. Compare with Zhou's proposal (50 hot peers)

Compare Figure 9&10 with Figure 7&8, the sharp of the broken lines almost remain the same because the data change is relatively small, but we can see form the upper-left table that much more skewed request distribution make original Chord suffer much more imbalanced load (local JFI from 0.2183 drops to 0.1983). While our proposal performs great even under highly skewed condition, local JFI from 0.9885 drops to 0.9751 which still remain highly balance. Local JFI in Zhou's proposal drops from 0.6851 to 0.6320 which also preforms good.

2) Global load balance: To evaluate the global load balance, we dispose the data by sorting the routing load of each peer in increasing order. Curve with smaller slope has more balanced routing load.

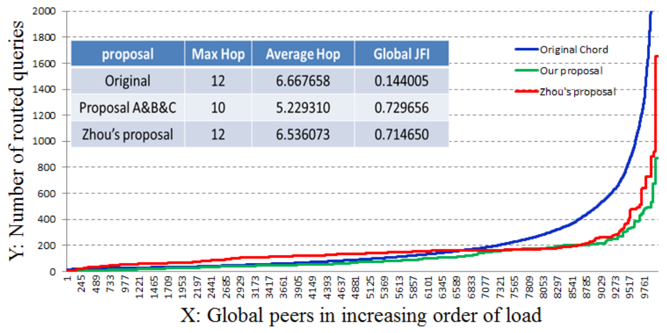
In Figure 11, the global JFI of our proposal is 0.8739 which is better than Zhou's 0.8670 and original Chord 0.2486. Even under much more highly skewed condition, our proposal still keeps the JFI 0.7297 in Figure 12 as the best one.

We can see the curve of our proposal is the most flat one with the least slope meaning our proposal produce better routing load fairness. As routing load increasing, the curve of original Chord becomes more and more steep and out of Y-axis range. But our proposal stops around 800, Zhou's proposal stops around 1600. In a much more skewed condition, original Chord becomes much steeper and not stable. But Zhou's and our proposal almost remain the same sharp proving that Zhou's and our proposal are stable even in much more skewed condition. The green line (our proposal) is always under the red line (Zhou's proposal) means that green line has smaller slope, our proposal is better.



proposal	Max Hop	Average Hop	Global JFI
Original	12	6.633696	0.248635
Proposal A&B&C	10	5.138649	0.870389
Zhou's proposal	13	6.400700	0.866981

Figure 11. Global view of routing load in increasing order (500 hot peers)



proposal	Max Hop	Average Hop	Global JFI
Original	12	6.667658	0.144005
Proposal A&B&C	10	5.229310	0.729656
Zhou's proposal	12	6.536073	0.714650

Figure 12. Global view of routing load in increasing order (50 hot peers)

Figure 16 is the entire overlay performance. Local JFI is the JFI of the local area of the hottest peer.

Original proposal suffer from imbalance routing load not only the local area of the hottest peer but also the entire network, and much more skewed request distribution even worsen its case. Proposal A improves the local and global routing load fairness in some degree. After applying proposal B, the global JFI has increased but the local JFI decrease slightly because of other hot peers' infection just as we analyzed in Section IV-B. Proposal C makes a qualitative leap that the local JFI almost reach *1* and global JFI stands great. After combining proposal A, B and C as our final proposal, we can see it is the best, both on local and global routing load balance and has a better query performance.

Entire overlay performance with 500 hot peers

proposal	Max Hop	Average Hop	Local JFI	Global JFI
Original	12	6.633696	0.218313	0.248635
Proposal A	11	6.106355	0.381119	0.336608
Proposal A&B	11	6.226049	0.378772	0.370122
Proposal A&B&C	10	5.138649	0.988463	0.870389
Zhou's proposal	13	6.400700	0.685075	0.866981

Entire overlay performance with 50 hot peers

proposal	Max Hop	Average Hop	Local JFI	Global JFI
Original	12	6.667658	0.198296	0.144005
Proposal A	11	6.119709	0.362910	0.212716
Proposal A&B	11	6.283191	0.361997	0.234835
Proposal A&B&C	10	5.229310	0.975112	0.729656
Zhou's proposal	12	6.536073	0.632025	0.714650

Figure 13. Entire overlay performance

VII. CONCLUSIONS

In this paper, we analysed the designing issues causing imbalanced routing load on Chord and external factors (like non-uniform request distribution and churn) that aggravate the imbalance. We propose three proposals each tackle on relevant issue and make a theoretical analyse after each of them. These three proposals are combined as our final proposal. To make the evaluation we apply our proposal under skewed and much more skewed request distribution condition. The simulation compare our proposal with original Chord and paper [8] by analysing routing load distribution on local and global area and Jain's Fairness Index of routing load and also the number of max and average hops. The results show that skewed request distribution does worsen the routing load fairness but our proposal works stable and the result basically follows our theoretical analysis in Section IV. Our proposal improve routing load balance far more balanced than the original Chord and better than paper [8] and also has a better query performance.

Several aspects of this study can be improved with further efforts. We only simulate our proposal under one external factor highly skewed request distribution, but the other external factor that aggravates the imbalanced routing load is churn, a common phenomenal that peers continuously join and leave the network. Evaluating the proposal under churn can be one of the future studies, and also the maintenance cost should also be quantificationally evaluated.

ACKNOWLEDGEMENT

This research was partially supported by Strategic Information and Communication R&D Promotion Program (SCOPE) of Ministry of Internal Affairs and Communications, Japan, Grant number: 122304003

REFERENCES

[1] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, et al., "Chord: a scalable peer-to-peer lookup protocol for Internet applications," Networking, IEEE/ACM Transactions on, vol. 11, pp. 17-32, 2003

[2] FIPS 180-1.Secure Hash Standard. U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, VA, Apr. 1995.

[3] M. Bienkowski, M. Korzeniowski, F. M. a. d. Heide, and F. M. Heide, "Dynamic Load Balancing in Distributed Hash Tables," in Proc. IPTPS, 2005, pp. 217-225.

[4] P. B. Godfrey and I. Stoica, "Heterogeneity and load balance in distributed hash tables," in INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, 2005, pp. 596-606 vol. 1.

[5] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Widearea cooperative storage with CFS," SIGOPS Oper. Syst. Rev., vol. 35, pp. 202-215, 2001

[6] R. Cuevas, M. Uruena, and A. Banchs, "Routing Fairness in Chord: Analysis and Enhancement," in Proc. INFOCOM 2009, IEEE, 2009, pp. 1449-1457.

[7] G. Cordasco and A. Sala, "2-Chord Halved". In Proc. of 2nd International Workshop on Hot Topics in Peer-toPeer Systems, (HOT-P2P 2005), pages 72–79, Jul. 2005.

[8] You Zhou and Koyanagi, K. "Improving routing load fairness in Structured P2P overlay networks," Advanced Communication Technology (ICACT), 2013 15th International Conference, 2013, Page: 724 - 728.

[9] P. B. Godfrey and I. Stoica, "Heterogeneity and load balance in distributed hash tables," in INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, 2005, pp. 596-606 vol. 1.

[10] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer filesharing workload," SIGOPS Oper. Syst. Rev., vol. 37, pp. 314-329, 2003.

[11] R. Jain, W. Hawe and D. Chiu. "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems". DEC-TR-301, 1984.

[12] A. Montresor and M. Jelasity, "PeerSim: A scalable P2P simulator," in Proc. Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference on, 2009, pp. 99-10

Lirong LIN
Master course student currently enrolled in the Graduate School of Information, Production and Systems (IPS), Waseda University, Japan. A member of the Thinking Networks Laboratory of IPS. Acquired the bachelor's degree majoring software engineering in Wuhan University, China in 2013. Main field of interests includes distributed systems, P2P networks, cloud computing and algorithm theories.

Keiichi KOYANAGI
Born in Tokyo, Japan on May 14, 1951. Received B.S. and M.S. degrees from Keio University, Tokyo, Japan, in1975 and 1977, respectively, and a Ph.D degree from Osaka University, Osaka, Japan in 1997. Since he joined NTT in 1997, he had been engaged in R&D of wide range of digital switching systems. Since 2003, he has been a professor of Waseda University, Graduate School of Information, Production, and Systems. He is a senior member of IEEE, and member of IPSJ, ACM.

Takeshi TSUCHIYA
Born in Shizuoka, Japan on Nov. 1978. He received his B.S. and M.S. degrees from in 2001 and 2003 from Tokyo University of Science, Japan Advanced Institute Science and Technology, and a Ph.D degree from Waseda University, Tokyo, Japan in 2009. He has been a assistant professor of Tokyo University of Science, Suwa, faculty of business administrations and information. His research interests are in the area of distributed systems and Cloud computing. He is a member of IEEE, IECE, and IPS

Tadashi MIYOSAWA
Born in Nagano, Japan on May. 1955. He received his B.S. degrees in 1979 from Department of Mathematics Waseda University, and a Ph.D degree from Waseda University, Tokyo, Japan in 2009. He has been a associate professor of Tokyo University of Science, Suwa, faculty of business administrations and information. His research interests are in the area of Multimedia computing and system. He is a member of IEEE, ITE, and IPSJ.

Hiroo HIROSE
Born in Tochigi, Japan on Oct. 1963. He received his B.S. degrees from in 1989 from Tokyo University of Science, Applied Mathematics, and a Ph.D degree from Daito Bunka University, Tokyo, Japan in 2010. He has been a professor of Tokyo University of Science, Suwa, faculty of business administrations and information. His research interests are in the area of Management Information Systems and e-Learning. He is a member of AACE, IPSJ, and JSISE.