Quasi-Chord: physical topology aware structured P2P network

SUN Mingsong [1] ZHANG Zhongqiu [2]

[1] Network Information Center, Harbin University of Science and Technology, China
[2] College of Computer Sci&Tech, Harbin University of Science and Technology, China

Abstract

A new approach is proposed in our paper to solve the topology mismatch problem in P2P network called Quasi-Chord. Quasi-Chord is in the light of the ideology of Chord. Our model utilizes global network position (GNP) system to coordinate the host on the physical layer and uses the Cantor space filling Curve to map the 2-dimensional geometrical space into 1-dimensional. Then we build up the Quasi-Chord model according to the Cantor value. Simulating experiment shows that this method can effectively lower the network delay and decrease the network flow.

Keywords: topology aware; peer to peer network; network coordinate; space filling curve

1. Introduction

In recent years, P2P technology has become the focus in the internet research area. Compared with other network model, P2P network has decentralized, scalable, robust, high performance, and other notable features, which makes P2P technology and its application concerned. Many collaborative applications are being built over P2P networks.

A p2p system is an overlay network built over a physical network. Each node is connected to a set of neighbors. However, the neighborhood of the node is set up without much concern with the underlying topology, which causes a mismatch between the P2P overlay and the physical network. This problem causes high latencies and communication overheads.

Approaches have been brought out to solve the P2P topology mismatch. The most representative of the solution is landmark clustering [1]. Landmarks are several fixed nodes in the systems. And every other node measures its round trip time from these landmarks. And sort the landmarks according the round trip time into a sequence. It uses this landmark sequence to represent the position of the node in the system. The nodes which have similar sequence are close to each other. Landmark clustering is a rough estimation of the position of the nodes. It can not help with the situation when the nodes are very close to each other.

Another approach to solve the topology mismatch is heuristic-based search. It is like searching method in a P2P system. It searches to get delay information instead of contents. This method can give us more detailed information about the physical layer. But the overhead of this method is too large. The blind searching will be too expensive for topology aware. The benefit of the method will be trade off.

In this paper we will propose a new approach to solve the P2P topology mismatch. We will construct an overlay network on the basis of the underlying topology. According to the ideology of Chord, we construct Quasi-Chord to realize topology aware in P2P network. The

experiment results proved the feasibility of this method, and showed that Quasi-Chord can enhance the efficiency of routing and reduce the network traffic.

2. Introduction to Chord

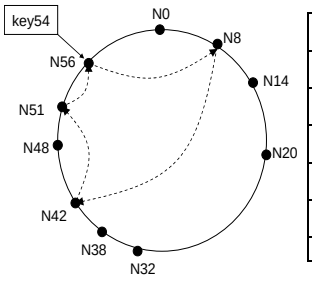
Because we will quote many definitions in Chord [2], in this section we introduce Chord first. Chord is a distributed lookup protocol proposed by MIT. The protocol locates the node that stores a particular data item in peer-to-peer applications.

According to Chord, each node is responsible for saving some index information for other nodes to lookup. Chord uses consistent hashing to provide fast distributed computation of a hash function mapping keys to nodes that responsible for them. The consistent hashing function assigns each node and key an m-bits identifier using a base hash function such as SHA-1. A node's identifier is chosen by hashing the node's IP address, while a key identifier is produced by hashing the key. The function of SHA-1 is mapping arbitrary length characters into 160 bits binary. The 160 bits binary will serve as the node's identifier in Chord. Then put the addresses in the ascending order to construct an identifier circle. Key k is assigned to the first node whose identifier is equal to or follows k in the identifier space. This node is called the successor of the key k, denoted by successor (k). If let m be the number of bits in the key/node identifiers. Each node, n, maintains a routing table with m entries, called the finger table. The i^{th} entry in the table at node n contains the identity of the first node, s , that succeeds n by at least 2^{i-1} on the identifier circle, $s = successor(n + 2^{i-1})$, where $1 \le i \le m$, We call the node s the i^{th} finger of node n, and denote it by n.finger[i].node.

The following table shows the definition of variables for node n using m-bit identifiers, which will continue use in Quasi-Chord model. And Figure 1 shows the Chord identifier circle and the finger table of node 8.

Table 1: definition of variables

Notation	Definition
Finger[k].start	$(n+2^{k-1}) \bmod 2^m$, $1 \le k \le m$
.interval	(finger[k].start, finger[k+1].start)
.node	First node $\ge$ n.finger[k].start
Successor	The next node on the identifier circle; finger[1].node
Predecessor	The previous node on the identifier circle



start	node
N8+1	N14
N8+2	N14
N8+4	N14
N8+8	N21
N8+16	N32
N8+32	N42

Fig. 1: Chord and finger table of node 8

3. Quasi-Chord

A problem that confronts Chord is that it does not consider the underlying topology when it constructs the identifier circle. To deal with this problem, we put forward the Quasi-Chord model. Quasi-Chord constructs the logic topology on the basis of the underlying topology information. So the nodes close in the underlying layer will be also close in the overlay network, which can greatly reduce the network traffic.

To build up Quasi-Chord model, we need three steps. The first step, we use GNP [3][4] to get each host's coordinate in P2P geometric space; The second, We use Cantor space filling curve [5] to convert the 2-dimensional space into 1-

dimensional; the last step, we will construct Quasi-Chord circle according to the host's Cantor value. In the following sections, we will describe each step in detail.

3.1. Compute hosts' coordinates with GNP

In this step, we position all the hosts in P2P network with GNP, and compute each host's coordinate. We model the P2P network as a geometric space with a well defined distance function. And characterize the position of any host in the internet by a point in this space such that the distance between any two hosts can be predicted with high accuracy by the output of the distance function evaluated on the hosts' coordinates.

To efficiently map Internet hosts to points in a geometric space, the key technique is to first compute the coordinates of a small distributed set of cooperating hosts called Landmarks based on the inter-Landmark distance. The Landmarks' coordinates serves as a frame of reference with which the coordinates of any ordinary host (relative to the Landmarks' coordinates) can then be derived based on the host's distances to the Landmarks.

I) Landmark Operations

Suppose we model the whole P2P network as a geometric space S.

Let us denote a host H in S as C_H^S, the distance function that operates on these coordinates as $f^S(.)$, and the computed distance between hosts H_1 and H_2, i.e. $f^S(C_{H1}^S, C_{H2}^S)$, as d_{H1H2}^S.

First we choose N landmarks L_1 to L_N. The landmarks simply measure the inter-Landmark round-trip times using ICMP ping messages and take the minimum of several measurements for each path to produce the bottom half of the N × N distance matrix, which is symmetric along the diagonal. Using the measured distances, the landmark computes the coordinates in S. The goal is to find a set of coordinates $C_{L1}^S \ldots\ldots C_{LN}^S$, for the N landmarks such that the overall error between the measured distances and the computed distances in S is minimized. That means we minimize the function $f_{obj_1}(.)$:

$$f_{obj_1}(C_{L_1}^S, \cdots C_{L_N}^S) = \sum_{L_iL_j \in \{L_1 \cdots L_N\} \| i>j} \varepsilon(d_{L_iL_j}, \overline{d}_{L_iL_j}^S) \quad (1)$$

where $\varepsilon(.)$ is an error measurement function, which is:

$$\varepsilon(d_{H_1H_2}, \overline{d}_{H_1H_2}) = \left(\frac{d_{H_1H_2} - \overline{d}_{H_1H_2}^S}{d_{H_1H_2}}\right)^2 \quad (2)$$

II) Ordinary Host Operations

Each ordinary host now measures its round-trip times to the N landmarks using ICMP ping messages and takes the minimum of several measurements for each path as the distance. An ordinary host computes its own coordinates that minimize the overall error between the measured and the computed host-to-landmark distances. That is to minimize the function $f_{obj_2}(.)$:

$$f_{obj_2}(C_H^S) = \sum_{L_i \in \{L_1 \cdots L_N\}} \varepsilon(d_{L_iH}, \overline{d}_{L_iH}^S) \quad (3)$$

where $\varepsilon(.)$ is the same as the previous one.

And other details about GNP please reference to their papers [3] and [4].

So after this first step, we can get each host's coordinate (x, y) in the P2P space.

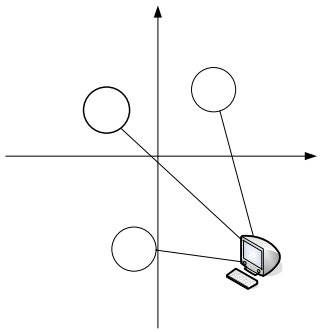
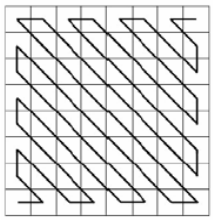


Fig. 2: Host's coordinate in P2P space

3.2. Map 2-dimentional space to 1-dimentional with Cantor SPF

Now we have got each host's coordinate(x,y), and the whole P2P space is considered to a 2-dinentional geometric space. In this step, we adopt Cantor space filling curve to convert 2-dimentional space into 1-dimentional.



35	36	48	49	57	58	62	63
21	34	37	47	50	56	59	61
20	22	33	38	46	51	55	60
10	19	23	32	39	45	52	54
9	11	18	24	31	40	44	53
3	8	12	17	25	30	41	43
2	4	7	13	16	26	29	42
0	1	5	6	14	15	27	28

Fig. 3: Cantor Chart when c=8

From Figure 3, we can have an intuitive view of the principle of Cantor SFC. According to the coordinate of each host, we can compute its Cantor value with Cantor Algorithm. First of all, we compute the diagonal k where the host located, and the diagonal k divided the Grid into two triangle parts, which is an upper triangular part and a lower part. In the upper part, remark the upper right corner to be the origin point and the whole upper triangular part to be a new coordinate system. Next we re-calculate the host's coordinate and its diagonal k. While the lower part, it need not to be recomputed. Then we compute k(k-1)div2 in both upper part and lower part to get the number of the hosts that the k-1 diagonal covered. At last, we determine the parity of k.
The detailed algorithm is as following:
Algorithm_Cantor_XY_to_N (x, y, c, N):

Input： x, y：location of a given node
c：size of domain (c × c)
Output： N：the serial number of the specific node

```
Begin
    K=x+y-1;
    {the node is located at the k-th diagonal from the origin}
    If （k>c） then
        { the node is located at the upper triangular part}
        X=c-x+1;
        {redefine the node's location in a new coordinate system}
        Y=c-y+1;
        K=x+y-1;
        {the k-th diagonal counted from the upper right}
        Qty1=k × (k-1) div 2；
        {quantity of nodes in the upper right of the k-th diagonal}
        Nodes=c × c;
        {quantity of nodes in the domain}
        If odd(k) then
          N=nodes-qty1-x；
          {done}
        Else  N=nodes-qty1-y；
          {done}
    Else { the node is located at the main diagonal or at the lower triangular part }
        Qty2=k × (k-1)div2；
        {quantity of nodes in the lower left of the k-th diagonal }
        If odd(k) then
            N=qty2+x-1;
            {done}
        Else
            N=qty2+y-1；
            {done}
End;
```

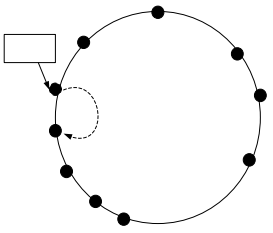
3.3. Build up Quasi-Chord model

Because hosts whose Cantor value are similar will be close to each other in the physical layer, we will build up the Quasi-Chord circle based on the Cantor value so that the identifier circle would aware the physical topology, which makes the Quasi-Chord topology awared.

Suppose c=max{｜x｜,｜y｜}, so the size of grid will be C × C, which is also the maximum number of hosts. Sort the hosts in ascending order to construct a big circle. Suppose n is the number of the hosts, and map the key to m bits identifier, where m meets $\log_2 n \le m \le \log_2(C \times C)$. Instead of hashing IP address to get the identifier of the node. We use the Cantor value to identify the node. Key k is assigned to the first node whose identifier is

equal to or follows k in the identifier space. In Quasi-Chord, each node stores 2 finger tables. One of them is clockwise finger table, which is similar to the one in Chord, while the other one is counterclockwise. This is because the first host in the circle and the last one are the farthest to each other. In Chord, the successor of the last node may be the first one. But in Quasi-Chord, this will bring heavy traffic to the circle. So the successor of the last host will never be the first one. So that no matter under what circumstance, a host can find its nearest successor.

Now we will discuss how to build the finger tables in detail. The clockwise table is similar to the one in Chord, but we will make sure that finger[i]. node is not less than the host ID itself. While in the counterclockwise table, finger[i]. node is not more than the host ID. And use finger[k]. start = $(n-2^{k-1})$ mod 2^m to replace finger[k].start = $(n+2^{k-1})$ mod 2^m in clockwise table. When a node lookups a key, it compare the key with its own identifier first, if the key is larger than the host's ID, it will lookup in the clockwise finger table, and otherwise, it will lookup the counterclockwise one. For example, the finger table of Node 51 is shown in Figure 4. If the key equals to 46, first the node compare 47 with 51, it is smaller than 51, so it will lookup in the counterclockwise finger table. Then it will send message to the node 48 directly. We can see that the counterclockwise finger table is very important in Quasi-Chord model. It brings much convenient and greatly reduce the traffic when we located the key.



Clockwise FT			Counterclockwise FT		
start	interval	node	start	interval	node
N52	[52,53)	N56	N50	(49,50]	N51
N53	[53,55)	N56	N49	(47,49]	N51
N55	[55,59)	N56	N47	(43,47]	N48
N59	[59,0)	N0	N43	(35,43]	N48
			N35	(19,35]	N38
			N19	(0,19]	N20

Fig. 4: Quasi-Chord and the finger table of N51

4. Experiment results

In Quasi-Chord model, link control and the amount of information transmission are similar to Chord, which won't have much impact on the overall performance of the system. But we have an algorithm that maps 2-dimensional space to 1-dimensional, which causes a little extra overhead in time complexity. In the following section, we will have some discussion on the effectiveness and efficiency of Quasi-Chord.

There are two main factors that measure the topology match of logic network and underlying physical network, which are: (1) the ratio of logic communication delay and physical unicast communication delay from source node to destination node. The less the ratio is, the closer between logical and physical distance are. At the same time, it dedicates the logical optimization does not equal to physical optimization. (2) the amount of copies of a single packet in the physical link. In the real link transmission, if a packet is copied too many times, it will occupy too much bandwidth. And if the replica increases sharply, the bandwidth will be exhausted. In this paper, by the simulation experiment, we compared Quasi-Chord and Chord with the above two aspects under the same experimental environment. And we analyzed the effectiveness and efficiency by the curve diagram. We used a discrete event simulator to simulate the number of nodes in the system as well as the landmark nodes and host nodes selection. Through the algo-

rithm described above to achieve the es-

The experiment environment was completed on the computer with XP operating system, AMD Athlon 64 X2 4000 + processor, 1G memory. In the experiment, we used INET to generate the underlying physical network topology, the underlying physical network consists of 10, 000 analog nodes, and the top P2P logical network consisted 60, 500, 1000, 2000, 3000, 4000, 5000 nodes in separate experiment. Figure 5 Figure 6 shows the results.

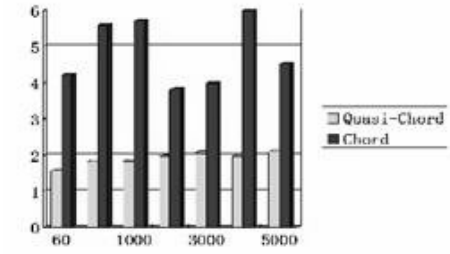


Fig. 5: Relative Delay Penalty

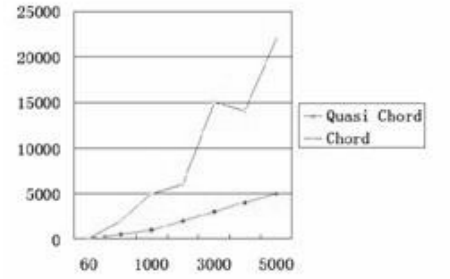


Fig. 6: Total Stress

As shown in Figure 5, with the growth of the number of nodes, the ratio of logical and physical communication delay from source node to destination node changes. It can be seen from the chart that the performance of Quasi-Chord is better than Chord. Quasi-Chord is more stable than Chord, and the overall ratio is lower. This indicates that the Quasi-Chord algorithm has a lot of improvement. Figure 6 describes the times of replicating a single packet. We can see that the Quasi-Chord algorithm is close to the linear growth, while in the Chord algorithm, the number tablishment of the Quasi- Chord model. of copied packets increases sharply when the number of nodes in the top layer increases, which seriously affected the network bandwidth. From the above chart, the original P2P is verified by the two comparative analysis, the more nodes join the network the better to show their superiority.

5. Conclusion

In response to the topology mismatch problem existing in structured P2P network, we propose a topology aware structured p2p network Quasi-Chord. The fundamental idea of Quasi-Chord is constructing top logical layer according to the topology of the underlying physical layer. So that the nodes close to each other in physical layer are also close in logical layer. Compared with the existing methods, our method has two main advantages: 1) The Correspondence relationship of physical layer and logic layer are more precise. 2) Combine network positioning and space filling curve to construct the p2p network, which can greatly reduce the network traffic brought by the TTL.

In the future research we will continue to improve the Quasi-Chord model to further enhance the efficiency of resource locator.

6. References

[1] S Ratnasamy, M Handley, R Karp, S Shenker, “Topologically-aware overlay construction and server selection,” [C] //. *In Proceedings of INFOCOM 2002. New York, NY, USA, June* 2002.

[2] I Stoica, R Morris, D Karger, M Kaashoek, H Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” [C] //. *In Proceedings of SIGCOMM 2001.*

San Diego, CA, USA, August, pp.149-160, 2001.
[3] T. S. Eugene Ng and Hui Zhang, "Predicting Internet Network Distance with Coordinates Based Approaches," [C] //. *Proceeding of the IEEE INFOCOM* 2002, c2002.
[4] T. S. Eugene Ng and Hui Zhang, "Towards Global Network Positioning," *Extended Abstract, ACM SIGCOMM Internet Measurement Workshop 2001, San Francisco, CA, November* 2001.
[5] Pei Zhi Lee, Chong Wei Huang, Tian Yuan Shih, "On Cantor's Space-Filing Curve," *Journal of Taiwan Geographic Information Science* (4), pp. 13-26, 2006.