Node Swap: A Universal Mechanism to Make P2P Overlay Topology-aware

Tongqing Qiu, Mao Ye and Guihai Chen
State Key Laboratory of Novel Software Technology,
Nanjing University, China
{*qtq,yemao*}*@dislab.nju.edu.cn, gchen@nju.edu.cn*

Abstract

P2P computing has emerged as a popular model for utilizing information and resource in Internet. It is built upon an overlay whose topology is independent of the underlying physical network. Because of the mismatch between the overlay and physical network, small number of logical hops may result in a long delay and excessive traffic. In this paper, we propose a universal mechanism-"node swap" to make the overlay topology be aware of the physical network. It is a generic, adaptive approach applicable for both unstructured and structured P2P overlay. Node swap is a heuristic and localized mechanism without much overhead, and simulation results show it can alleviate the mismatch problem of P2P overlay efficiently.

1. Introduction

In P2P network, each node provides service to other participating nodes as well as receives services from them. According to the organization of P2P overlay topology, the current P2P architecture can be classified to two types: unstructured P2P (e.g. Gnutella[1]) and structured P2P (e.g. Chord[2], CAN[3]). In unstructured P2P, the overlay topology is often random and unstructured mesh. The query is flooded through this mesh hop-by-hop till success/failure; in structured P2P, the overlay topology is some structured topology, such as ring, mesh and hypercube. The query is executed hop-by-hop through the structured topology, and is sure to be successful after some deterministic hops under ideal case.

P2P system is built upon an overlay network whose topology is independent of the underlying physical network. The neighborhood of each node in P2P overlay is nondeterministic, due to the arbitrary organization of unstructured P2P or the hash-based property of structured P2P. A well-routed message path in an overlay network with a small number of logical hops may result in a long delay because of the undesirably long distance in some physical links. This mismatch problem between the overlay and physical network is a great barrier to build an effective large-scale P2P overlay network. In this paper, we propose a generic adaptive mechanism-"node swap" to resolve the mismatch problem existed in P2P overlay. Regardless the type of the P2P network, using localized information probing and node swap, we enable the P2P overlay topology aware of the underlying physical networks. To the best of our knowledge, this mechanism is the first one which is effective in both unstructured and structured P2P systems.

The remainder of this paper is organized as follows. In section 2, we will introduce the universal mechanism in detail and analyze the property of it. Section 3 shows the simulation results of node swap in P2P systems. In section 4, we give a concise review about several most related works. Finally, we conclude this paper in section 5.

2. The Universal Mechanism

The overlay network can be modelled by a directed graph $G = (V, E)$, where V is the set of nodes in the network, and E is the set of links between nodes. An edge (u, v) in E means that u knows a direct way to send a message to v. If $(u, v) \in E$, we call that node u is the *predecessor* of node v, and node v is the *successor* of node u. The one is a *neighbor* of the other. For unstructured P2P systems, the links are bidirectional,

Notation	Meaning
V	the set of all nodes in the system
E	the set of all edges in the overlay network
t_0	the time before nodes u and v swap
t_1	the time after nodes u and v swap
$N_{t_i}(u)$	the neighbor set of node u
$d(i,j)$	latency between nodes i and j
L_{t_i}	the accumulated latency value of overlay
AL	the average latency of the overlay

Table 1. The notation table

which means that if $(u,v) \in E$, then $(v,u) \in E$. But it is not correct in structured P2P systems. That is why we use the directed graph not the undirected one to represent the overlay network.

2.1. Node Swap

In order to explain our approach in detail, we gives several useful notations for our expression in table 1. We assume there is a potential swap between node u and v. t_0 and t_1 represent time before and after a swap respectively. The neighbor set of node u is defined as follows:

$$N(u) = \{i | i \in V \wedge ((u,i) \in E \vee (i,u) \in E)\} \qquad (1)$$

In order to simplify the neighbors' processing, we extend the routing table by recording not only successor nodes but also predecessor ones[1]. The size of extended routing table is about twice as large as the size of the original one. In some symmetrical systems like Gnutella or CAN, there is even no increase. At the beginning of adjustment, each node u gets its neighbor list $N_{t_0}(u)$ and the initialized latency information $\sum_{i \in N_{t_0}(u)} d(u,i)$. After the initialization, the node u will periodically contacts a random node v in a fixed time interval T. Node v is k hops away from node u. The TTL-packet is used to realize this contact. Then nodes u and v exchange their initialized latency information and address lists. Both of them calculate the latency information $\sum_{i \in N_{t_1}(u)} d(u,i)$ and $\sum_{i \in N_{t_1}(v)} d(v,i)$. Then they exchange their local latency information and calculate the variable $Diff$ independently.

$$\begin{aligned} Diff = \quad & \sum_{i \in N_{t_0}(u)} d(u,i) + \sum_{i \in N_{t_0}(v)} d(v,i) \\ & - \sum_{i \in N_{t_1}(u)} d(u,i) - \sum_{i \in N_{t_1}(v)} d(v,i) \end{aligned} \qquad (2)$$

If $Diff \leqslant 0$, it means that a swap can not gain any benefit. So no operation is performed. If $Diff > 0$, nodes u and v will do swap operation. They exchange routing tables and identifiers [2] respectively. Besides, both of them will notify their predecessors to change the routing tables and recalculate the initialized sums. As the routing table is extended, the notification can be realized directly. Even if there is no extension of the routing table, exchange of two nodes can be realized using a series of *leave()* and *join()* procedures. However, because these procedures lead to complicated reconstruction operations, we do not use it.

2.2. Property Analysis

In this subsection, we exhibit the topology maintainability and generic characteristic of node swap theoretically. Further, we analyze the effectiveness of node swap to demonstrate its capability in resolving mismatch problem of overlay network.

Theorem 1. *Let graph $G(V,E)$ denote the network overlay at time t_0 before swap. Without loss of generality, we assume node v_x and v_y do swap operation during the period t_0 to t_1. The deduced graph $G'(V',E')$ at time t_1 is isomorphic to graph $G(V,E)$ at time t_0.*

Proof. First, it is clear that $|V| = |V'|$ and $|E| = |E'|$. We can construct a one-one mapping between V and V'. The symbol $\longleftrightarrow$ is used to represent the one-one mapping relation

- For $\forall i \neq x \wedge i \neq y$, $v_i \in V \longleftrightarrow v'_i \in V'$
- For x and y, $v_x \in V \longleftrightarrow v_y \in V'$, $v_y \in V \longleftrightarrow v_x \in V'$

V_1 is used to denote the set of un-swapped nodes and V_2 presents the set of swapped nodes. Subsequently, we construct a one-one mapping between E and E':

[1] As a matter of fact, most structured P2P systems selectively record several predecessor nodes in order to improve the ability of fault resilience.

[2] Identifiers are only in structured P2P systems.

- For $\forall v_i, v_j \in V_1$, $e_{ij} \in E \longleftrightarrow e'_{ij} \in E'$
- For $\forall v_i \in V_1$, $v_j \in V_2 (v_j = v_x \vee v_j = v_y)$, we have that $e_{ix} \in E \longleftrightarrow e'_{iy} \in E'$, $e_{iy} \in E \longleftrightarrow e'_{ix} \in E'$. Similarly, $e_{xi} \in E \longleftrightarrow e'_{yi} \in E'$, $e_{yi} \in E \longleftrightarrow e'_{xi} \in E'$.
- For $\forall v_i, v_j \in V_2$, we have that $e_{ij} \in E \longleftrightarrow e'_{ji} \in E'$

Observing the mappings we have constructed, it is easy to conclude that G is isomorphic to G'. □

Corollary 1. *After an arbitrary serial of swap operation, the ultimate deduced graph is isomorphic to the original graph.*

Proof. Observing the theorem 1, it is easy to deduce the result to a more general case - an arbitrary serial of swap operation. □

The above theoretical analysis exhibits the wonderful property of node swap mechanism. It keeps the connectivity of network and also maintains the topology invariable. Therefore, this mechanism is suitable for different topologies. In addition, it preserves the anonymity in P2P systems. So it guarantee the security and safety in a certain degree.

To explain the effectiveness of node swap, we make several definitions and explain the meaning of notations first. We define *stretch* as the ratio of the average logical link latency over the average physical link latency. Stretch is a common parameter to quantify the topology match degree. *Average latency (AL)* is a basic parameter to quantify the property of a network. If there are n nodes in a network, then[3]

$$AL = (\sum_{i \in V} \sum_{j \in V} d(i,j))/n^2 \quad (3)$$

We analyze the change of average latency after a swap between nodes u and v. Supposing that the number of nodes is invariable during $t_0 \rightarrow t_1$, so the *accumulated latency* (L_{t_i}) is analyzed instead. Next two equations show this change:

$$L_{t_0} = C + \sum_{i \in N_{t_0}(u)} \alpha_i d(u,i) + \sum_{i \in N_{t_0}(v)} \beta_i d(v,i) \quad (4)$$

[3]We assume the latency between one node and itself is zero.

$$L_{t_1} = C + \sum_{i \in N_{t_1}(v)} \gamma_i d(v,i) + \sum_{i \in N_{t_1}(u)} \delta_i d(u,i) \quad (5)$$

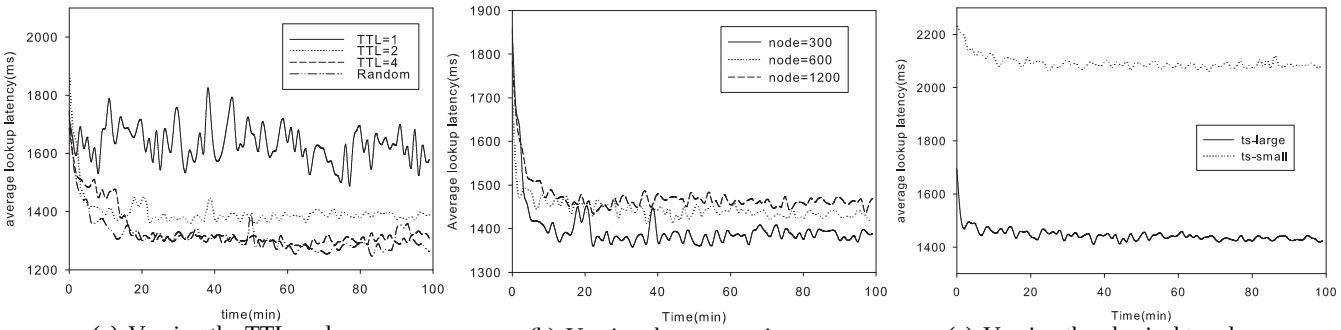
In equation 4 and 5, C represents the invariable part before and after one swap operation. The coefficients of the summations $\alpha, \beta, \gamma, \delta$ represent the times each neighbor link used. We notice that nodes u and v just exchange their neighbors, so $N_{t_1}(v) = N_{t_0}(u)$ and $N_{t_0}(v) = N_{t_1}(u)$. Besides, assuming that each link has the same probability to be visited, then $\alpha_i \approx \gamma_i$ and $\beta_i \approx \delta_i$. To calculate the variation by $(4) - (5)$, we get that if $Diff > 0$ then $L_{t_0} > L_{t_1}$, which implies that a swap makes the stretch reduced. It is worth to mention that it is an approximate analysis. In fact, when the positions of the nodes changed, the times each neighbor link visited are variable. In other words, those coefficients are different, that is why not all swaps can reduce the average latency. We will see that in our experiment.

3. Simulation Methodology

We use the GT-ITM topology generator [4] to generate transit-stub models of the physical network. In fact, we generate two different kinds of topologies. The first topology, *ts-large* has 70 transit domains, 5 transit nodes per transit domain, 3 stub domains attached to each transit node and 2 nodes in each stub domain. The second one, *ts-small*, differs from ts-large in that it has only 11 transit domains, but there are 15 nodes in each sub domain. Intuitively, ts-large has a larger backbone and sparser edge network than ts-small. Except in the experiment of physical topology, we always choose ts-large to represent a situation in which the overlay consists of nodes scattered in the entire Internet and only very few nodes from the same edge network join the overlay. We also assign latencies of 5, 20 and 100ms to stub-stub, stub-transit and transit-transit links respectively. Then, several nodes are selected from the topology as overlay nodes, with the node number $n = \{300, 600, 1200\}$. Gnutella and Chord are chosen as the platform of our simulation. All parameters used in the experiments and their default type/value are in table 2

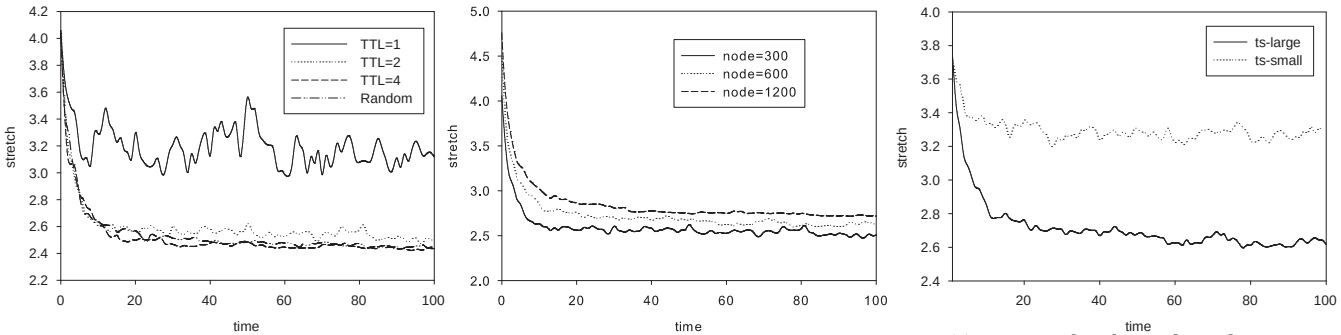
Parameter	Meaning	Default type/value
ts	the transit-stub model	ts-large
n	the number of nodes in Gnutella/Chord	600
TTL	time to live for probing	2
T	the time interval for swap	1 min
δ	the percentage of nodes join and leave	0
t	the time interval δ% nodes join and leave	1 min

Table 2. Parameter selection



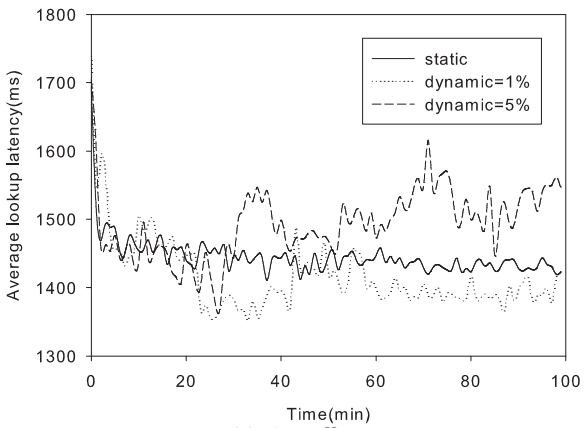
(a) Varying the TTL scale (b) Varying the system size (c) Varying the physical topology

Figure 1. Effectiveness of node swap in Gnutella-like environment

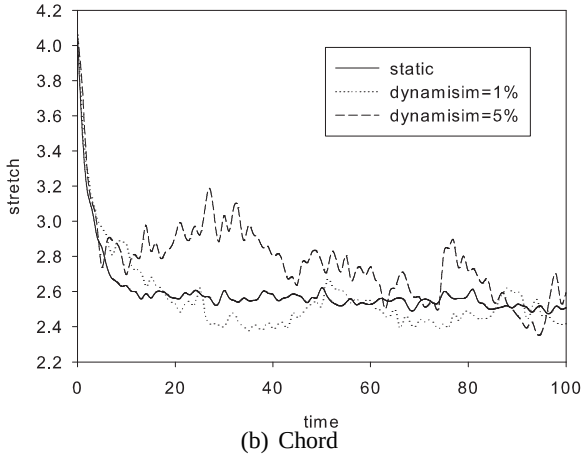


(a) Varying the TTL scale (b) Varying the system size (c) Varying the physical topology

Figure 2. Effectiveness of node swap in Chord environment



(a) Gnutella



(b) Chord

Figure 3. Effectiveness in dynamic environment

3.1. Effectiveness of Swap

Stretch is common metric to characterize the match degree of the overlay to the physical topology in structured P2P systems. However, in unstructured P2P systems, messages are sent by flooding method, so it is not practical to calculate the latency between any two nodes. As a result, we choose average lookup latency in Gnutella instead. Both stretch and average lookup latency vary according to the time. The time interval T is fixed as one minute. Figure 1(a) 2(a) show the impact of the TTL scale in two different systems. We choose node number $n = 600$ and four typical scenes of probing node. In a centric scene, we can just choose a random node as the probing target. In a distributed system, we use $TTL = \{1,2,4\}$. $TTL = 1$ means probing neighbors; $TTL = 2$ means probing neighbors' neighbors and $TTL = 4$ means probing the node half of diameter away from the original node[4]. We can find that neighbors' swap is not suitable as it can't greatly reduce the stretch, while other three different ways have nearly the same impact on stretch reduction. The reason is obvious, as $TTL = 1$ gets only neighbor information which is too limited. Given that random probing is not practical in a distributed system, only when $TTL \geq 2$ can achieve a good performance in a P2P system. In order to minimize cost, $TTL = 2$ may be a better choice, and it will be used in next several experiments. In figure 1(a) 2(a), we can also discover that the stretch is not reduced all the time, which is consistent with our approximate analysis in section 2.2.

Figure 1(b) 2(b) illustrate the impact of system size. We choose $n = 300, 600, 1200$. The effectiveness is reduced as the size becomes larger. This situation can be explained in two different directions. First, when the system has a large size and probing method fixes TTL as 2, the information we get is relatively limited. Second, as we choose the nodes from the same physical network, when the overlay becomes larger, it is closer to the physical topology. So the efficiency will be not so obvious.

The impact of physical topology is presented in figure 1(c) 2(c). We have generated two different types of topologies *ts-large* and *ts-small* by GT-ITM tools. Both of them contain 2200 nodes. It is obvious that ts-large topology has much better performance. In ts-large topology, only a few stub nodes attach to transit nodes. So the probability that two stub nodes belong to different transit nodes is relatively high. Accordingly, the probability that these nodes exchange is also great. It means that two *far* nodes make adjustment to match the physical topology with high probability (*w.h.p.*). This kind of swap will greatly improve the performance of the system. As we mentioned above, ts-large topology is much like the Internet, so our method will significantly improve performance in a real large-scale system.

[4]As node number is 600, we suppose that the diameter $d = \log_2 n \approx 8$ in Chord.

3.2. In Dynamic Environment

Dynamism is a very important property in P2P systems. In this part, we try to discover the impact of dynamism on our approach. Although people do several searches about dynamism of the P2P system [5], there is not a standard model to describe it. In our simulation, we just set a very simple dynamic environment. There are δ percent of nodes join and δ percent of nodes leave at a time interval t. $\delta = \{0, 1, 5\}$ and $t = 1min$. Figure 3 shows the results. It is obvious that stretch fluctuates greatly when the system is under a dynamic situation in which 5 percent of nodes change per minute. The situation is worse in Gnutella network. The original overlay of Gnutella can reflect some proximity information because each new node will add nearby nodes as neighbors. When nodes arrive and depart, this kind of proximity information is lost. So the dynamic change affects the unstructured P2P systems more obviously. However, we can see that nodes' arrival and departure may not lead the system to a poor match degree. It's possible that nodes' changes have the similar effect as our swap operation which reduces the stretch of the system. Besides, the effectiveness of our mechanism in dynamic environment is relative to two time intervals T and t. If t is much smaller than T, we will get similar results as in static environment. It is obvious that we should make a tradeoff between the effectiveness and overhead. In a word, although there is a fluctuation, our method can be still effective in dynamic environment.

4. Related Work

As topology aware issue has been extensively exploited for several years, several interesting methods are proposed to solve the mismatch problem of overlay network. Here we review a few most related ones in the literature. LTM[6] is a location-aware topology matching scheme proposed for unstructured P2P systems. In LTM, each peer issues a detector in a small region so that the peers receiving the detector can record relative delay information. Based on the delay information, a receiver can detect and cut most of the inefficient and redundant logical links and add closer nodes as its direct neighbors. LTM is a typical method which is only applicable for Gnutella-like overlay network. SAT-Match[7] is a topology-aware mechanism for the system CAN. The basic operation in the protocol is "jump". When one node discovers several nearby nodes by flooding, it will jump to the nearest node in the nearby area. However, the arbitrary jump violates the anonymity of the P2P system, node is easily attacked by a hacker. In addition, although the author mentioned the impact of the dynamism, we can not find detailed evaluation in different dynamic environments.

5. Conclusion

This paper proposes a universal mechanism to solve the mismatching problem in P2P systems. This mechanism is totally protocol-independent, which can be easily used on any P2P systems. It is effective in a dynamic environment. Due to the limitation of the paper size, we omit further improvement of node swap method. Fox example, we have some adaptive methods to reduce the overhead of probing. Beside, we also found that our method can further improve performance when combining with other topology-aware method. Readers may refer [8] to get more information.

References

[1] http://rfc-gnutella.sourceforge.net.

[2] 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 the ACM SIGCOMM*, 2001.

[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proceedings of the ACM SIGCOMM*, 2001.

[4] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee., "How to model an internetwork," in *Proceedings of INFOCOM*, 1996.

[5] Z. Ge, D. R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling peer-to-peer file sharing systems," in *Proceedings of IEEE INFOCOM*, 2003.

[6] Y. Liu, L. Xiao, X. Liu, L. M. Ni, and X. Zhang, "Location awareness in unstructured peer-to-peer systems," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 16, no. 2, pp. 163–174, Febuary 2005.

[7] S. Ren, L. Guo, S. Jiang, and X. Zhang, "SAT-Match: A self-adaptive topology matching method to achieve low lookup latency in structured P2P overlay networks," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS04)*, 2004.

[8] T. Qiu and G. Chen, "A generic method to make structured p2p systems topology-aware," Tech. Rep., 2005.