Peer Clustering and Firework Query Model

Cheuk Hang Ng , Ka Cheung Sia
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong SAR
{chng,kcsia}@cse.cuhk.edu.hk

ABSTRACT

In this paper, we present a new strategy for information retrieval over the peer-to-peer (P2P) network. To avoid query messages flooding and saving resources in handling irrelevant queries in the P2P network, we propose a method for clustering peers that share similar properties together, thus, data inside the P2P network will be organized in a fashion similar to a Yellow Pages. In order to make use of our clustered P2P network efficiently, we also propose a new intelligent query routing strategy, the Firework Query Model. In contrast to broadcasting query message, our query message will first walk around the network from peer to peer randomly, once it reaches the target cluster, the query message explodes and is broadcasted by peers inside the cluster. We present our algorithm, give analysis and experimental results to demonstrate our method.

Keywords

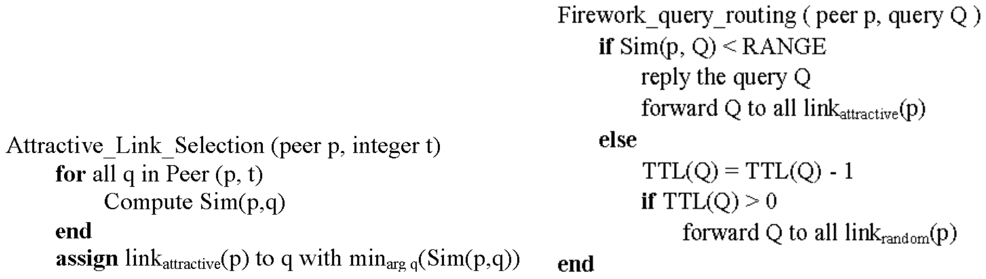
Peer-to-Peer, Information Retrieval, Peer Clustering, Intelligent Query Routing

1 Introduction

The appearance of P2P applications such as Gnutella [1] and Napster [2] has demonstrated the significance of distributed information sharing systems. Users can access the information distributed among peers around the world. These models offer advantage of decentralization by distributing the storage, information and computation cost among the peers. However, since there is no centralized server to keep track of what data are being stored in what peer, we do not know which peer contains the information we want. Therefore, when a peer wants to search for a file, he needs to broadcast query to all his peers. Likewise, his peers propagate the query to their peers and so on.

There are two problems associated with current strategies. Since every query is broadcasted to every peer in the network, each peer has to waste resources in handling irrelevant query. Moreover, broadcasting query messages across the network also increases network traffic. Portmann et. al. [3] investigated the problem of scalability in P2P network due to network traffic cost. Rowstron et. al. [5] and Stoica et. al. [6] proposed their algorithm targeting at reducing network traffic in P2P network. Ramanathan et. al. [4] proposed a clustering strategy to reduce query traffic, while increasing the goodness of search result.

In order to solve the problems above, our proposed strategy will cluster peers of similar properties together, just like the organization in a Yellow Pages, which makes query more systematic. In addition, our intelligent Firework Query Model will help in reducing network traffic by forwarding query selectively rather than the original Brute-Force-Search (BFS) broadcasting. Under our query model, irrelevant query subjects to a particular peer will not go into its cluster, once the query arrives at a matching peer, it is broadcasted inside its cluster. As a result, we avoid unnecessary traffic while fully utilize each query message.



```
Attractive_Link_Selection (peer p, integer t)
    for all q in Peer (p, t)
        Compute Sim(p,q)
    end
    assign $link_{attractive}(p)$ to q with $min_{arg\ q}(Sim(p,q))$
```

```
Firework_query_routing ( peer p, query Q )
    if Sim(p, Q) < RANGE
        reply the query Q
        forward Q to all $link_{attractive}(p)$
    else
        TTL(Q) = TTL(Q) - 1
        if TTL(Q) > 0
            forward Q to all $link_{random}(p)$
end
```

Figure 1. Attractive Link Selection (a) left and Firework Query Algorithm (b) right

2 Peer Clustering and Firework Query Model

2.1 Peer Clustering

Inside our P2P network, each peer is connected by two types of link, **random** link and **attractive** link. Before continuing on how to manipulate these two types of link, we introduce the following terms:

- Random link - $link_{random}(p)$: The connection which a peer p makes randomly to another peer in the network. It is chosen by the user.
- Attractive link - $link_{attractive}(p)$: The connection which a peer p makes explicitly to another peer, which they share similar data.
- $Cat(p)$,$Cat(Q)$: A signature value representing the characteristic of a peer p, and a query Q respectively.
- $Sim(p, q)$: The similarity between two peers p, q, which is a distance measure between $Cat(p)$ and $Cat(q)$.
- $Sim(p, Q)$: The similarity between a peer p and a query Q, ,which is a distance measure between $Cat(p)$ and $Cat(Q)$.
- $Peer(p, t)$: The set of peers which a peer p can reach within t hops.

Based on the above definition, we introduce an algorithm to choose the attractive link as illustrated in Fig. 1(a). Referring to Fig. 2(a), we illustrate how our peer clustering strategy works. In the beginning, let us assume every peer can be represented by a value $Cat(p)$ based on the characteristic of that peer p respectively. In the mean time, we consider this value to be geographical location. When a peer joins the network, it will connect to another peer randomly. Through ping-pong messages [1], it learns the set of peers within a certain number of hops away from it ($Peer(p, t)$). Based on the attractive link algorithm, it will connect to a peer that shares similar properties through attractive links. In Fig. 2 (a), the peer CN_4 first connects to UK_3 by random link, and by ping-pong messages, it learns the location of other peers belong to category CN, then it makes an attractive link to CN_2 to perform peer clustering. Our algorithm will choose a similar peer with largest hop count to connect, so CN_2 is chosen. As a result, you notice that peers of similar characteristic will connect together to form a cluster, which makes information retrieval much more systematic.

In the example we used, geographical information is used to determine the similarity between peers. However, our algorithm is generic in the sense that, as long as we can choose a good descriptor to describe a peer, for example, the major file type it shares, the images' content information it shares, will also work properly.

2.2 Firework Query Model

To make use of our clustered P2P network, we propose the Firework Query Model. In this model, a query message will first walk around the network from peer to peer by random link, then once it reaches the target cluster, the query message will be broadcasted by peers using attractive link insides the cluster as shown in Fig. 2 (b).

To decide which peer a query is being sent to, we introduce an algorithm to determine when and how a query message explodes in Fig. 1 (b). Referring to Fig. 2 (a) again, we illustrate an example to show how the firework query model works. Assume the new peer CN_4, whose geographical location is China, initiates a search to find an American song. Since the geographical information of his query is far away from his geographical location, therefore, CN_4 will send the query to UK_3 through random link. When UK_3 receives the query, it will forward the query to US_2 through random link again. After walking around the network randomly, the query message reaches its target cluster and starts to explode. During the explosion, US_2 will broadcast the query to US_1 through attractive link. Likewise, US_1 will broadcast the query to US_2, CN_1 and so on.

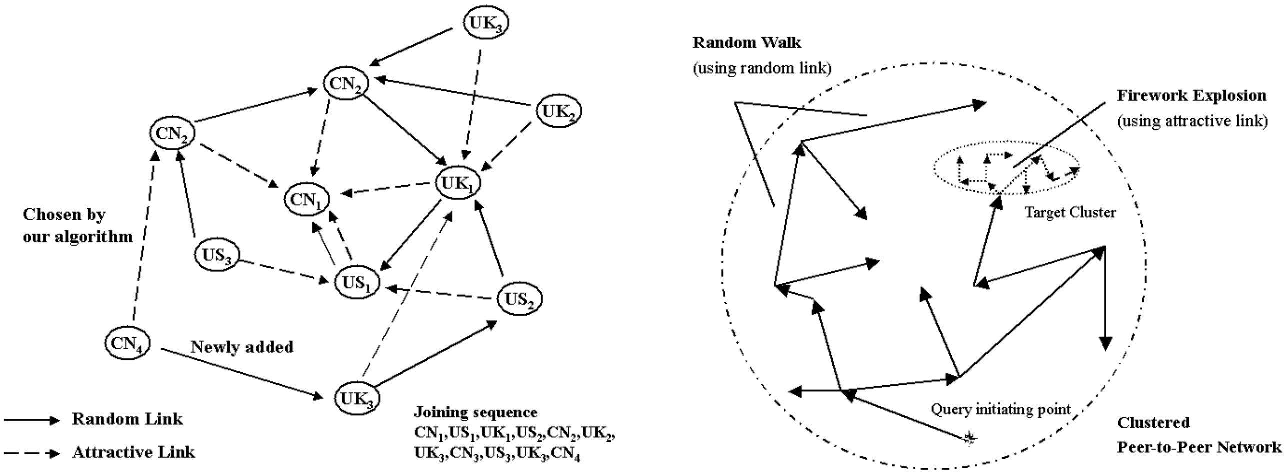


Figure 2. Illustration of peer clustering (a) left and firework query (b) right

Similar to IP packets and Gnutella messages, our query messages also have a Time-To-Live (TTL). This avoids messages from circling around the network forever. Once the TTL decreases to zero, the message will be dropped and no longer forwarded. The only difference between our query messages and the gnutella messages is that TTL will not be decreased when the messages are sent through attractive link. The reason is that, when a query reaches its target cluster, all peers inside the cluster are highly related, in order to get as more hits as possible, there is no reason to decrease the TTL and prevent further searching inside the highly related cluster.

3 Experiment

We develop a program to simulate the P2P environment and verify the effectiveness of our proposed strategy. We investigate the effect on query efficiency subjects to increasing number of peers, different values of query message's TTL. We formulate five cases to compare the difference in performance subject to different network set-up methods and query strategies, described in the following.

- Random BFS (BFS): Like pure P2P network (Gnutella) [1], network topology is built randomly and queries are broadcasted to all connected peers.

- Centralized fuzzy (CF): Peer acquires knowledge of other peers from a centralized server to determine how to make attractive link. Queries are forwarded to attractive link if $Sim(p, q) < \epsilon$, otherwise to random link. We consider this to be fuzzy because queries are considered similar when the similarity measures are still within a range. In our experiments, for example, category 5 peer will treat category 4 and 6 queries as matching queries and forward them through attractive link.

- Learning fuzzy (LF): Peer acquires knowledge of others through ping-pong messages to learn a partial picture of the network, and establishes attractive link based on this information. Queries are forwarded in the same way as Centralized fuzzy.

- Centralized discrete (CD): Network setting-up method is the same as Centralized fuzzy. Queries are forwarded to attractive link if $Sim(p, q) = 0$, otherwise to random link.

- Learning discrete (LD): Network setting-up method is the same as Learning fuzzy. Queries are forwarded in the same way as Centralized discrete.

In the experiments, we assume there are ten categories of peers, and assign different $Cat(p)$ values evenly. Also, queries generated only fall in these ten categories, when a peer receives a query of the same category, it will fire a hit count. We measure the recall of a particular query as the number of hit counts over total number of peers under that category. We record three measures, the number of query message, recall, and recall per query message (efficiency of query) against the number of peers and TTL. Referring to Fig. 3(b), our proposed strategy shows a promising sub-linear increases in number of query messages with increasing TTL, while BFS increases exponentially. In addition, we show that our strategy has much improvement on recall per query message than traditional BFS as shown in Fig. 3(b) and 4(b), also, the recall performance is shown in Fig. 4(a), which indicates our algorithm still outperforms BFS while we use less number of query messages. Detailed data are listed in Table 1 and 2.

4 Conclusion and Future Work

We empirically examine several aspects of the performance of clustered P2P network. Among the many possible ways to extend the current work, perhaps the most challenging one is to choose one or more descriptor(s) to describe a particular peer for use in clustering, where this may be a single value, a vector or even multi-dimension vectors to precisely describe a peer.

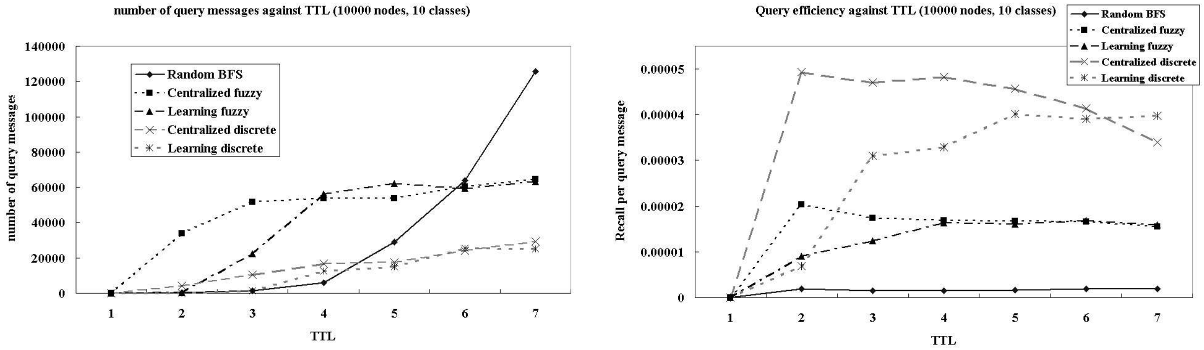


Figure 3. Number of query packet (a) left and query efficiency (b) right against TTL

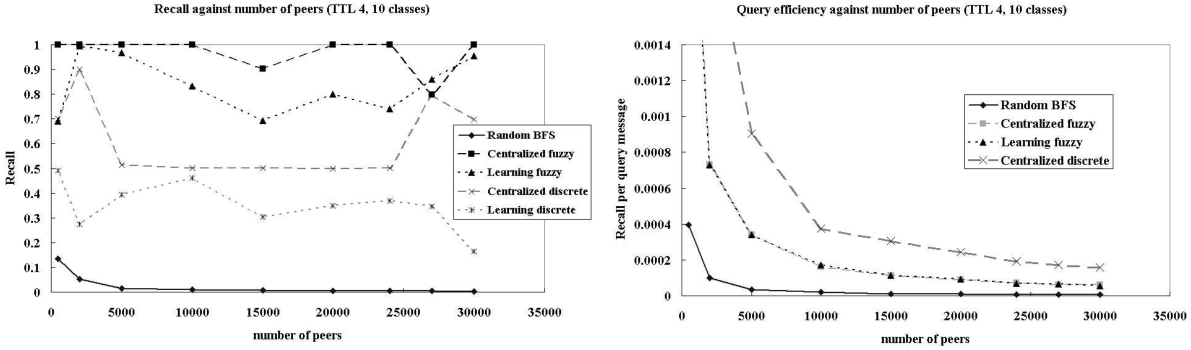


Figure 4. Recall measure (a) left and Query efficiency (b) right against number of peers

Table 1. Average number of query messages and recall per query message against increasing TTL over 10 simulation runs

	number of query messages						recall per query message ($\times 10^{-6}$)				
TTL	BFS	CF	LF	CD	LD		BFS	CF	LF	CD	LD
2	272	33986	191	4039	**173**		1.845	20.382	8.967	**49.366**	6.896
3	**1245**	51781	22337	10612	1500		1.508	17.374	12.365	**46.984**	30.994
4	**6053**	53635	56235	16593	12632		1.506	16.847	16.281	**48.211**	32.951
5	28959	53825	62263	17496	**14895**		1.557	16.733	16.061	**45.688**	40.219
6	63896	60504	59419	**24179**	25566		1.877	16.528	16.830	**41.358**	39.114
7	125749	64548	63070	29476	**25162**		1.842	15.492	15.855	33.926	**39.742**

Table 2. Average recall and recall per query message ($\times 10^{-6}$) against increasing number of peers over 10 simulation runs

	Recall						Recall per query message ($\times 10^{-6}$)				
number of peers	BFS	CF	LF	CD	LD		BFS	CF	LF	CD	LD
30000	0.003407	**1.000000**	0.954259	0.699008	0.163576		6	61	55	**158**	143
27000	0.005353	0.798179	**0.861528**	0.797958	0.346832		5	61	66	**172**	138
24000	0.004764	**1.000000**	0.741765	0.500960	0.368099		8	71	70	**194**	160
20000	0.004273	**1.000000**	0.799729	0.499800	0.349756		9	88	92	**244**	204
15000	0.008146	**0.902563**	0.693782	0.501639	0.304515		10	113	113	**307**	236
10000	0.009970	**1.000000**	0.833564	0.500250	0.461937		19	162	174	**376**	340
5000	0.016126	**1.000000**	0.967871	0.513788	0.395052		33	342	340	**905**	737
2000	0.052970	**1.000000**	0.993896	0.899050	0.275470		101	731	729	**2076**	1198
500	0.134000	**1.000000**	0.689796	0.699797	0.490239		395	2890	2815	**5598**	3954

5 Acknowledgements

This project is supported in part by grants from the Hong Kong's Research Grants Council (RGC) under CUHK 4407/99E and #2050259.

References

[1] The gnutella homepage. In *http://www.gnutella.com.*

[2] The napster homepage. In *http://www.napster.com.*

[3] M. Portmann, P. Sookavatana, S. Ardon, and A. Seneviratne. The cost of peer discovery and searching in the gnutella peer-to-peer file sharing protocol. In *Proceedings to the International Conference on Networks*, volume 1, 2001.

[4] M. K. Ramanathan, V. Kalogeraki, and J. Pruyne. Finding good peers in peer-to-peer networks. In *International Parallel and Distributed and Computing Symposium*, April 2002.

[5] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms*, November 2001.

[6] 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 ACM SIGCOMM*, pages 149–160, August 2001.