Topology-aware Multi-cluster Architecture Based on Efficient Index Techniques

Yun He[1], Qi Zhao[1], Jianzhong Zhang[2], Gongyi Wu[2]

Department of Computer Science and Technology, Nankai University,
Tianjin 300072, China
[1]{hey1630, qizhao6688}@mail.nankai.edu.cn
[2]{zjz, wgy}@nankai.edu.cn

Abstract. In this paper, we focus on how to construct an efficient unstructured P2P system. The main contributions of our proposal are two-fold. First, aiming at alleviating the topology mismatch problem between the P2P logical overlay network and the physical underlying network, we proposed a Topology-aware Multi-cluster Overlay (TMO) architecture where peers self-organize into clusters based on network locality. Second, in order to further improve the search efficiency of the TMO architecture, we present two novel index techniques, namely, cluster-index technique and topic-index technique. The two different techniques are highly effective in different application domains in which the TMO architecture is deployed. The simulation results indicate that our proposed schemes are efficient in both resource usage and data retrieval.

1 Introduction

In recent years, there has been much interest in peer-to-peer (P2P) systems because they provide a good substrate for building large scale data sharing and content distribution applications. P2P systems can be broadly classified into two categories: unstructured and structured P2P systems.

Unstructured P2P systems, like Gnutella [1] and KaZaA [2], organize peers in a random graph and use flooding on the graph to query documents stored at overlay peers. The floods support arbitrary queries, but are not scalable because they cause exponentially increased network traffic. In contrast, structured P2P systems are developed to perform key queries by constructing Distributed Hash Tables (DHTs), such as Chord [3], CAN [4], and Pastry [5], etc. Although such schemes provide good performance for exact match queries, they almost don't work for range, approximate, or text queries. Thus, many agree that unstructured P2P systems are more suitable for mass-market file sharing applications.

In traditional unstructured P2P systems, the mechanism of a peer randomly joining and leaving causes topology mismatch between the P2P logical overlay network and the physical underlying network [6]. This topology mismatch problem causes a large amount of unnecessary traffic, which brings great stress on the Internet infrastructure.

The objective of this paper is to construct an efficient unstructured P2P system. We propose an application architecture called Topology-aware Multi-cluster Overlay (TMO), which has two levels. Peers in the lower level self-organize into clusters based on network locality, aiming at alleviating the topology mismatch problem. The clusters are organized into the upper level overlay defined by a directed graph (e.g. DTH graph) such that the efficient routing between clusters can be easily achieved.

In order to further improve the search efficiency of the TMO architecture, we present two novel index techniques, namely, *cluster-index technique* and *topic-index technique*. In the cluster-index technique, each cluster has content indices from all peers of some other clusters. When a query is submitted, full search scope can be achieved even though some of the clusters are directly probed. In the topic-index technique, all the documents stored in the network are classified into topics. Each document's index is sent to the cluster responsible for the topic that the document belongs to. A query probes only a few clusters that have the largest number of results under a particular topic. The two different techniques are highly effective in different application domains in which the TMO architecture is deployed.

The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 describes the TMO architecture in details. Section 4 and Section 5 describes the cluster-index technique and the topic-index technique, respectively. In Section 6, the simulation results are presented, followed by conclusions in Section 7.

2 Related Work

There are several P2P systems that use indexing approaches. For example, Napster [7] is a centralized system that uses specialized peers to maintain the indices of the documents available in the overlay network. To find a document, the user queries an index peer to identify peers having documents with the content of interest. KaZaA [2] is a popular super-peer network where a super-peer acts as a centralized server to a subset of clients. In order to process queries for its clients, a super-peer keeps an index over its clients' documents.

Although original Gnutella does not build indices, some indexing approaches have been proposed to make Gnutella scalable. For example, in Local Indices policy proposed in [9], each peer indexes the files stored at all peers within a certain radius r and can answer queries on behalf of all of them. The work in [10] proposes 3 types of Routing Indices (RIs), namely compound RIs, hop-count RIs and exponential RIs to facilitate search in Gnutella. In particular, peers forward queries to their neighbors based on their own RIs. Ways to improve searching has been extensively studied using Search/Index Links (SIL) [8]. SIL points out that a parallel search cluster based P2P network is superior to a super-peer network for several important scenarios. However, the mechanism of how to break the P2P networks into multiple clusters has not been mentioned yet.

3 Topology-aware Multi-cluster Overlay

We begin by presenting a general framework for TMO. We assume that each participating peer has an IP address. The peers are organized into clusters. Each cluster has a unique cluster id. We let N denote the number of clusters, and C_i denote both cluster i and the id of cluster i. The clusters are organized by a graph (X, U), where $X=\{C_0, C_1, \ldots, C_{N-1}\}$ is the set of all clusters and U is a given set of virtual edges between the nodes (that is, clusters) in X. The edges in U may be unidirectional or not. We believe that DHT graphs [3, 4, 5] can be efficiently used as the graph (X, U). In this paper, we use the Chord DHT graph as an example.

TMO consists of two kinds of links. *Short-distance links* connect peers within a cluster. *Long-distance links* connect pairs of peers from different clusters. Two peers are *short-distance neighbors* if they are connected by a short-distance link. We require that if p_i is a peer in C_i, and (C_i, C_j) is a unidirectional edge in U, then p_i knows the IP address of a peer $p_j \in C_j$. With this knowledge, p_i establishes a long-distance link to p_j, and p_j becomes a *long-distance neighbor* of p_i. It is important to note that p_i keeps only one long-distance neighbor in each of C_i's neighboring clusters. In addition, if C_j is the successor node of C_i in Chord DHT graph [3], we will say that p_j is p_i's *first long-distance neighbor*. Of course, each peer has only one first long-distance neighbor, which plays the key role in the cluster-index technique that will be described in Section 4.

Figure 1 shows an example of TMO architecture. Four clusters C_0, C_1, C_2 and C_3 are organized by a Chord graph. Each peer in cluster C_0 selects a long-distance neighbor from both C_1 and C_2, because the neighboring clusters of C_0 are C_1 and C_2.

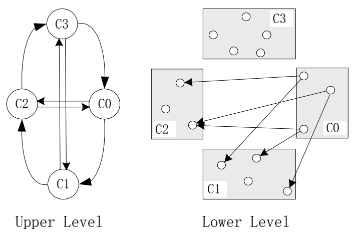

Fig. 1. A TMO architecture (N=4)

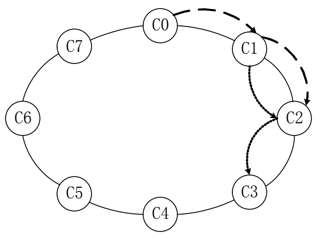

Fig. 2. A TMO-CI system (N=8, d=2)

3.1 TMO Construction

One key idea of TMO is that it partitions peers into clusters by network locality. We use the landmark clustering method proposed in [6] to generate topology information for clustering physically close peers. Landmark clustering method requires a set of well-known landmark nodes spread across the Internet. A peer measures the network-level Round-Trip-times (RTTs) to each of these landmark nodes and sorts the landmark nodes in terms of increasing RTTs. Peers with the same or similar landmark ordering are considered close to each other, and are expected to join the same cluster. The interested reader is referred to [6] for these details.

When a new peer wants to join a TMO system, it first measures RTTs to all landmark nodes to get a landmark ordering, which assigns it to a specific cluster. Then the new peer sends a JOIN message destined for the target cluster. The message is sent into TMO via any existing peer. Peers receiving the message use Chord routing mechanism to forward the message via long-distance links, until it reaches a random peer in the target cluster. A new peer can only join one cluster at the same time.

After the new peer has joined the target cluster, it gets short-distance neighbors in the Gnutella fashion, and gets long-distance neighbors as follows. If *x* is the new peer in cluster C_i, it will send request messages to its short-distance neighbors for the IP addresses of their long-distance neighbors. If *x* gets the IP address of another peer *y* belonging to cluster C_j, which is a neighboring cluster of C_i, then *x* will try to connect *y*. If the attempt succeeds, *y* will become *x*'s long-distance neighbor for cluster C_j, or else *y* will send the IP address of its short-distance neighbors to *x*, then *x* will try to connect these peers for long-distance neighbors. It is enough for *x* to keep only one long-distance neighbor in each of C_i's neighboring clusters. But in practice, *x* may cache more than one candidate peer in each cluster to improve system tolerance.

There are two reasons for why we use the Chord graph to organize clusters. First, using the Chord graph can maintain network connectivity and route queries in a few hops without requiring too many long-distance links per peer. Second, the Chord graph is able to embed the two index techniques that we will describe later.

4 Cluster-index Technique

In this section we present an efficient index technique: cluster-index technique (TMO-CI). Let us suppose that peer *x* constructs content index over its own documents soon after it joins a cluster. The content index, which is used to assist in answering queries, may be inverted lists of words, sets of metadata or simply a list of filenames. The peer *x* will send its content index to its first long-distance neighbor. The peer that receives *x*'s content index will cache the index and select its first long-distance neighbor to relay the index. The whole process is repeated until *d* different peers have received *x*'s content index, where *d* is a system-wide variable known as the *depth parameter*. If these *d* peers receive queries, they can process the queries on behalf of *x*. It is important to note that a peer does not send any content indices to other long-distance neighbors except its first long-distance neighbor.

4.1 Select the Directly Probed Clusters

Using the cluster-index technique, a cluster can be *directly probed* or *indirectly probed*. Figure 2 shows a TMO-CI system where the depth parameter *d* is set to 2. So each peer in cluster C_i sends its content index to its first long-distance neighbor in cluster C_{i+1} and in turn to a peer in cluster C_{i+2}. Hence, we can deduce that the cluster C_{i+2} has content indices of all peers in cluster C_i and C_{i+1}. If a query is propagated in cluster C_{i+2}, we will say that cluster C_{i+2} is directly probed, and will say that cluster C_i and C_{i+1} are indirectly probed. For a query, it is unnecessary to require a cluster to be directly probed if it has been indirectly probed already.

A probe to a directly probed cluster proceeds in two steps. First, the Chord routing mechanism in system's upper level makes sure that the query message is routed to the target cluster. Next, an intra-cluster flood mechanism is used to further propagate the query within the cluster. Although, we implement only the intra-cluster flood mechanism, the TMO system can also use other search mechanisms, such as Random Walks [11] or Gossip [12], to propagate queries within a cluster.

We propose that the selection of directly probed clusters should follow two criterions. First, with the same number of probed clusters, minimize the number of directly probed clusters, aiming to reduce query traffic. Second, make sure that the query messages sent from the source peer to these directly probed clusters traverse as few long-distance links as possible, aiming to shorten the query response time.

For example, illustrating in figure 2, if a peer in cluster C_0 submits a query and requires all of the eight clusters to be probed, then the cluster C_0, C_2 and C_5 are selected as the directly probed clusters according to our criterions.

5 Topic-index Technique

In this section, we present another index technique for the TMO architecture: topic-index (TMO-TI). In the TMO-TI system, all the documents stored in the network are classified into topics. For example, for a music sharing application, TMO-TI may create topics like "Rock", "Heavy metal", "Classical" and so forth. Each document belongs to one or more topics. For each topic, there are one or more clusters responsible for it. A cluster collects the indices of documents belong to specific topics that it is responsible for. An index of a document may be an inverted list of words or simply the name of the document. We let M denote the number of topics, T_i ($0 \leqslant i \leqslant M$-1) denote topic i, and S_i ($0 \leqslant i \leqslant M$-1) denote the set of clusters responsible for T_i. It is obvious that $S_i \subseteq \{C_0, C_1, \ldots, C_{N-1}\}$.

A peer will classify its own documents after it joins a cluster. If some documents do not belong to the topics that the peer's cluster is responsible for, the peer will send these documents' indices to the responsible clusters. To explain, we assume p is a peer in cluster C_i, for each document D_i stored on p, if $D_i \in T_i$ and $C_i \notin S_i$, then the index of D_i will be send to a cluster $C_j \in S_i$.

A query is also classified into one or more topics, and the clusters responsible for the topics will be directly probed. The classification of documents and queries can be done manually or automatically. However, classifiers may make mistakes by returning the wrong topics for a query or document. In the simulations we will study how much the system is affected in the presence of classifier mistakes.

6 Simulations

The two types of topologies, physical topology and logical topology are needed in the simulation. A transit-stub topology [15] of approximately 35,000 nodes is generated as the physical topology in which the delays of intra-transit domain links, stub-transit links and intra-stub domain links are set to 20, 5 and 2ms respectively. We generate a flat logical topology with average connectivity degree of 6 for measuring Gnutella search. This logical topology has 16,000 peers, each of which is uniquely mapped to one physical node. In order to measure our TMO search, we randomly select 8 physical nodes as the landmark nodes, and partition all the logical peers into 8-32 clusters based on locality.

We distribute 3,000 different documents of varying popularity in the simulation. A zipfian distribution is used to model both the replication distribution and the query distribution to achieve results similar to the results in [13]: The most popular 10% of documents amount for 50% of the total number of stored documents and account for over 50% of total queries. The documents are classified into 50 different topics, each of which only one cluster is responsible for.

The quality of a search mechanism is judged by the following metrics:

➢ Traffic cost: We define traffic cost as $\sum_{i=1}^{N_m} y_i s_i$, where N_m is the number of messages, s_i is the size of message i, and y_i is the delay of the link which message i traverses. Implicit here is the assumption that links with higher delay and messages with larger size tend to be associated with higher traffic cost.

➢ Hits: We define hits as the size of total result set for a query.

➢ Response Time: We define response time as the time that has elapsed from when the query is submitted by the peer, to when the peer receives the first result.

6.1 Results of Gnutella search

We conduct our simulations to evaluate the performance of TMO search against Gnutella search. In the first simulation, we examine the performance of Gnutella search with different TTLs. The simulation results in Table 1 indicate that increasing the TTL of Gnutella search increases the traffic cost quickly, but results in more hits and better response time as we model the network delay in the simulation.

Table 1. Results for Gnutella search

Scheme	Cost	Hits	Time		Scheme	Cost	Hits	Time
TTL=7	3275376	52.76	354		TTL=5	799886	24.75	404.8
TTL=6	1955845	43.49	393.2		TTL=4	183658	7.34	406

6.2 Effectiveness of TMO-CI

In this subsection, we examine the effectiveness of TMO-CI search with different TTLs (that is, TTLs of intra-cluster floods used within each cluster). Here we representatively present the results based on 16 clusters only, since changing the number of clusters produces similar results. We set the depth parameter d to from 0 (means the degenerate TMO case without cluster-index technique) to 7. When a query is submitted, we require all the clusters to be probed, aiming to get full search scope.

Figure 3, figure 4 and figure 5 show the average query traffic cost, query hits and response time, respectively. Based on these simulation results, we make the following inferences on TMO-CI search.

➢ Similar to Gnutella search, increasing the TTL of TMO-CI search also increases traffic cost, but results in more hits and better response time.

➢ Compared with Gnutella search, by appropriately selecting TTLs the TMO-CI search really reduces the average traffic cost while achieving the same or similar query hits. Furthermore, increasing the value of d usually results in larger reduction of query traffic cost. For example, when TTL= 5, the strategies that setting *d* to from 0 to 7 achieve similar query hits of Gnutella search with TTL=7, but reduce average traffic cost by from 38% to 90%. In fact, the traffic cost each query generates largely depends on the number of directly probed clusters, which is equal to $\lceil N/(d+1) \rceil$, where *N* is the number of clusters.

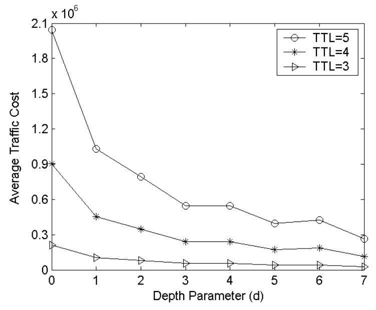

Fig.3. Average Traffic Cost in TMO-CI

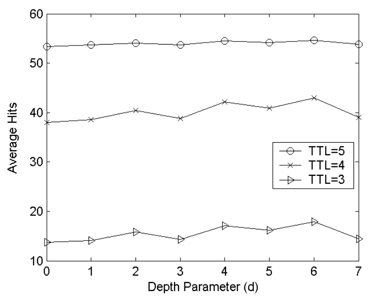

Fig.4. Average Hits in TMO-CI

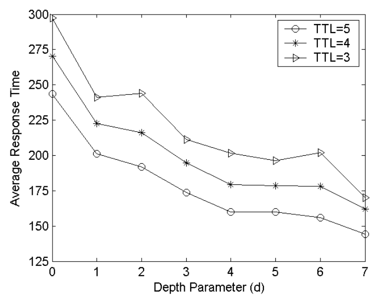

Fig.5. Average Response Time in TMO-CI

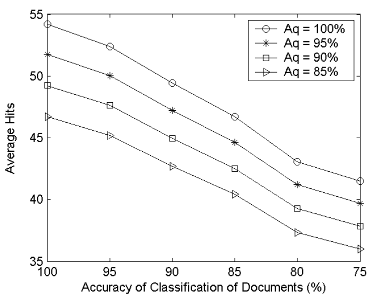

Fig.6. Average Hits in TMO-TI

- The TMO-CI search can shorten the response time, since it takes the physical network topology into consideration when the overlay is constructed. Besides, increasing the value of *d* usually results in better response time, since a peer can answer queries for many other peers when *d* is set to a large value. For example, compared with Gnutella search, the strategies that setting *d* to from 0 to 7 in TMO-CI search can shorten the response time by from 31% to 59%.
- It may be difficult to choose the appropriate TTL for TMO-CI search. Empirically, we would choose a smaller TTL when the number of clusters is large, and a larger TTL when in the contrast case. However, we also believe that it is difficult to choose the appropriate TTL for the Gnutella search.

6.3 Effectiveness of TMO-TI

In this subsection, we examine the effectiveness of TMO-TI search. We let A_d denote the accuracy of classification of documents, and A_q denote the accuracy of classification of queries.

We representatively present the results based on TTL=5 only. Figure 6 illustrates the average query hits of TMO-TI. Different curves correspond to the performance on different A_q with different value of A_d. If classifiers don't make mistake, the TMO-TI search achieves similar query hits compared with Gnutella search. However, with the decrements of A_q and A_d, the query hits also decreases. For example, TMO-TI search achieves 67% query hits of Gnutella search when A_q=85% and A_d=75%.

Figure 7 shows the average response time for TMO-TI search. We can see that decreasing the values of A_q and A_d results in a little longer response time.

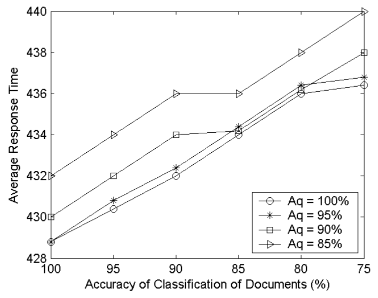

Fig.7. Average Response Time in TMO-TI

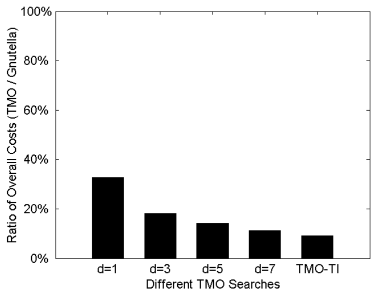

Fig.8. Ratio of Overall Traffic Costs

In the simulation, we find that TMO-TI search only generates around 5% of the average query traffic cost of Gnutella search. Changing the values of A_q and A_d has little influence on the average traffic cost that the queries generate.

6.4 The Impact of Index Update

One of the key factors that affect the performance of TMO system is the frequency of index update operations, which heavily depends on the dynamic nature of overlay network. In a real environment, the source peer should do index update operations periodically, which incurs extra traffic cost. Especially, in the TMO-CI system, increasing the depth parameter *d* could increase the extra traffic cost proportionally.

In the simulation, we assume that each peer executes 10 index update operations per minute. We also assume that each peer issues 0.3 queries per minute, which is calculated from the observation data shown in [14], i.e., 12,805 unique IP addresses issued 1,146,782 queries in 5 hours. Figure 8 shows the ratio of the overall traffic costs in different TMO systems to the overall traffic cost in Gnutella system. Compared with Gnutella system, our TMO-CI system can reduce overall traffic cost by at least 67%, and TMO-TI system can reduce overall traffic cost by 90%. Thus, the search improvements afforded by TMO and the two index techniques are seldom outweighed by the extra traffic cost of index update operations.

Based on the observations above, we believe that the strength of the cluster-index technique lies in that it can help the TMO system to reduce both query traffic cost and response time without decreasing the query hits. Thus, it is highly effective in the applications where 100% recall is required, for example, a patent information sharing application. The advantage of the topic-index technique is that it can help the TMO system to reduce a quite large amount of traffic cost, though it may result in reduction of query hits. Thus, it is highly effective in the applications where users are satisfied with tens of (but not all) results, such as the music sharing application.

7 Conclusion

In this paper, we proposed TMO, a topology-aware multi-cluster overlay architecture which using a hierarchical structure with two levels. Furthermore, we present two novel index techniques, namely cluster-index technique and topic-index technique that can be incorporated into the TMO system to enhance search efficiency. From our simulation results we conclude that TMO with index techniques offers significant improvements versus Gnutella-like overlay networks. We believe that the TMO system and the two index techniques can help improve the search performance of current and future P2P systems.

References

1. Gnutella. http://gnutella.wego.com/
2. KaZaA. http://www.kazaa.com/
3. I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," In *Proceedings of ACM SIGCOMM*, 2001.
4. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content -addressable addressable network," In *Proceedings of ACM SIGCOMM*, 2001.
5. A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," In *Proceedings of International Conference on Distributed Systems Platforms*, 2001.
6. S. Ratnasamy, N. Handley, R. Karp, and S. Shenker, "Topologically-Aware Overlay Construction and Server Selection," In *Proceedings of IEEE INFOCOM*, 2002.
7. Napster. http://www.napster.com/
8. B. F. Cooper and H. Garcia-Molina, "Studying search networks with SIL," In *Proceedings of IPTPS*, 2003.
9. B. Yang and H. Garcia-Molina, "Improving Search in Peer-to-Peer Networks," In *Proceedings of IEEE ICDCS*, 2002.
10. A. Crespo and H. Garcia-Molina, "Routing indices for peer-to-peer systems," In *Proceedings of 22nd International Conference on Distributed Computing Systems*, 2002.
11. C. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," In *Proceedings of ACM ICS*, 2002.
12. Kermarrec, A.-N., Massoulie, L., and Ganesh, A. J, "Probabilistic reliable dissemination in large-scale systems," *IEEE Transactions on Parallel and Distributed Systems*, 2003.
13. J. Chu, K. Labonte, and B. Levine, "Availability and Locality Measurements of Peer-to-Peer File Systems," In *Proceedings of SPIE*, 2002.
14. K. Sripanidkulchai, "The popularity of Gnutella queries and its implications on scalability," In *Proceedings of O'Reilly's Peer-to-Peer and Web Services Conference*, 2001.
15. E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to Model An Internetwork," In *Proceedings of IEEE INFOCOM*, 1996.