Ontology-Based Clustering and Routing in Peer-to-Peer Networks

Juan Li, Son Vuong
Computer Science Department, University of British Columbia
{juanli, vuong}@cs.ubc.ca

Abstract

How to improve the performance of content searching in peer-to-peer (P2P) systems is a challenging issue. In this paper we attack this problem by proposing a new decentralized P2P architecture – ontology-based community overlays. The system exploits the semantic property of the content in the network to cluster nodes sharing similar interest together to improve the query and searching performance. Specifically, a distributed hash table (DHT) based overlay is constructed to assist peers organizing into communities. Those peers in the same community form a Gnutella-like unstructured overlay. This architecture helps reduce the search time and decrease the network traffic by minimizing the number of messages propagated in the system. Moreover, it retains the desirable properties of existing unstructured architectures, including being fully decentralized with loose structure, and supporting complex queries. We demonstrate by simulation, that with this architecture, peers can get more relevant resources faster and with less traffic generated.

1. Introduction

With the growing volume of information and resources stored in P2P networks, it is becoming increasingly difficult to search for desired resources in P2P networks. Various querying and routing techniques have been proposed for searching in P2P networks. Currently, many popular commercial P2P applications [15, 26] are built on top of unstructured P2P networks. Flooding is the predominant search technique in unstructured networks. This method, though simple, does not scale well in terms of message overhead. Recent work on DHTs [16–19] captures the relationship between content name and content location. They map an object to a key and guarantee to locate the object within a bounded number of hops. However, a missing feature in these systems is the ability to support complex queries. More recently, a few studies [10, 21, 22] extend the DHT scheme to support keywords or multi-attribute queries. However they may incur either huge traffic load for result intersection, or large overhead for multiple publication and update.

In this paper, we propose a query routing approach which organizes nodes into community overlays according to different categories defined in nodes' content ontology. It forwards queries only to semantically related overlays, and thus alleviating the traffic load caused by flooding in large-sized Gnutella systems. Specifically, the system includes two types of overlay: an upper-level DHT-based category overlay and multiple lower-level decentralized unstructured community overlays. The DHT overlay helps peers with similar interests locate each other and form community overlays. In addition, it also helps node forward queries to right communities. A community overlay is composed of nodes with similar interest. It is responsible for resolving queries related to this interest. The benefit of this routing approach is that queries will travel less and success is more likely to be achieved with a smaller number of hops. As we show in this paper, the system exhibits many plausible characteristics, such as: supporting complex queries, fast response, low bandwidth, and robust behavior.

The remainder of this paper is organized as follows. Related work is discussed in Section 2. Section 3 describes the construction, maintenance and routing of the DHT and community overlay networks. Section 4 gives the experimental results. Section 5 concludes the paper.

2. Related work

Gnutella [15] is a representative of unstructured P2P systems. It uses flooding to locate and retrieve shared files. One advantage of it is inherent scalability and good fault tolerance. However, Gnutella faces serious scaling problems when the network is very large [4]. There have been numerous attempts to enhance its scalability. For example, [1] improves the efficiency of searches in unstructured P2P networks by topology adaptation, replication, and flow control. Yang and Garcia-Molina [2] present several strategies, such as iterative deepening and the DBF algorithm to reduce the overhead of searching. Another algorithm, called probabilistic flooding [3], has been modeled using percolation theory. Random Walks [14, 20] are an alternative to flooding for unstructured searches. They usually utilize previous experience to help forward the query. They can reduce the amount of network traffic, but it is at the cost of query latency. In addition, they are effective in locating popular content, but perform poorly for more rare content.

Recently, hierarchical super-peer systems [13] have been proposed to improve search efficiency. They utilize the heterogeneity existing in P2P networks and adopt hierarchy in the form of Ultrapeers (Gnutella [15]) or Super-nodes (FastTrack [12]). These powerful nodes maintain the indices for other nodes, therefore searching can be carried out only among these more powerful ones. The introduction of a new level of hierarchy in the system increases the scale and speed of a query lookup.

DHTs [16–19] have received a lot of attention in the last few years. They map a content identifier to a key, and guarantee that content can be located within a bounded number of hops. These systems have been shown to be scalable and efficient. However, a missing feature is keyword searching and support for more advanced queries. Another hurdle to DHT deployment is their tight control of both data placement and network topology, which makes them more sensitive to failures and “churn”.

More recently, A few studies extend the DHTs to address the problems mentioned above. In [21], DHTs have been proposed to implement multiple-keyword searching. The idea is to map each keyword to a key and publish links to objects based on these keys. A query with multiple keywords has to lookup each keyword and returns the intersection. All related peers have to exchange large amounts of data to get the intersection. Systems like [23] try to avoid this multiple lookup and intersection by storing a complete keyword list on each node. However, this may incur a huge overhead on publishing and storing the keywords. Skipnet [9] provides range queries and data placement flexibility on top of DHTs, but it requires many pointers, thereby increasing the maintenance traffic. And it cannot guarantee system-wide load balancing.

Semantic Web [5] attempts to define the metadata information model for the World Wide Web to aid in information retrieval and aggregation. It provides general languages for describing any metadata. Currently, many P2P applications [6, 7, 8] have leveraged semantic web technologies to add semantics to P2P systems, and thus improving the effectiveness of content and query representation, and the efficiency of content searching.

3. Framework

This section gives a detailed explanation of the system architecture. Particularly, it explains how to cluster nodes, how to create, maintain and search the community overlays.

3.1. Clustering policies

To create peer communities, we need a clustering policy to cluster peers. Since a peer’s interests can be represented by its local data, data ontology properties can be used to classify peers. Ontology is defined as “a formal, explicit specification of a shared conceptualization”, which can refer to the shared understanding of some domains of interests. The criteria of ontology classification can be very flexible. For example, it can be a global taxonomy like domains defined in Yahoo [24] and DMOZ [25]. Or it can be general ontology formalizing notions such as processes and events, time and space, physical objects, and so on. We can assume taxonomy is defined in the data ontology and each data item can be classified into one type. Also, we assume queries can be classified with the same policy.

3.2. Community construction

After the clustering policies have been determined, the system can cluster peers according to it and create peer communities. This section explains how to construct community networks in detail.

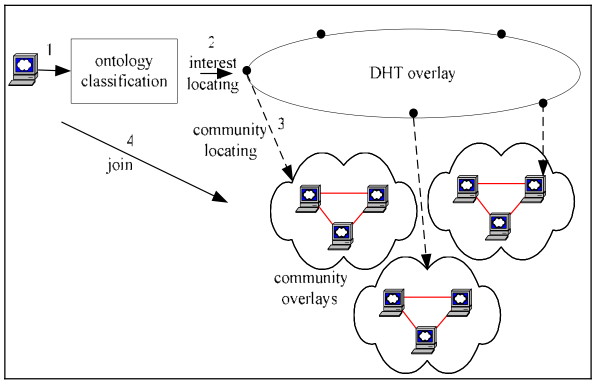

Figure 1. System architecture

As shown in Fig. 1, the system consists of two kinds of logical overlays: one DHT overlay and multiple community overlays. The function of DHT overlay is to assist the construction of community overlays. Nodes in the DHT overlay are also user nodes that are stable and have good Internet connectivity compared with the rest nodes. Excluding ephemeral nodes from the DHT overlay avoids unnecessary maintenance cost. A node joins the DHT overlay only when three conditions are satisfied: (1) The node satisfies the capacity requirements, i.e., it is powerful. (2) It is stable for a threshold time. (3) The load inside the DHT overlay is high. Only all these three conditions are satisfied, can a node join the DHT overlay to take over some load.

In this paper we use Chord [18] to implement the DHT overlay. Nodes are organized into a ring topology corresponding to an ID space ranging from 0 to 2^m-1, where m can be set according to the network size. Each node is assigned an ID drawn from this ID space, and is responsible for the key range between its ID and the ID of the previous node on the ring. When a new node joins the system, it first classifies its data, and then registers its major interests to the DHT overlay. Specifically, an interest of the joining node is hashed into a key in the ID space. The node in the DHT overlay, whose ID immediately follows the interest's key is responsible for that interest, and is called the access point of that interest. From the access point, the new node would know other existing nodes sharing the same interest. Then this new node contacts some of these existing nodes and joins their communities. In this way, nodes always connect to other nodes sharing common interests, and thus they can form a community overlay. Chord on average routes a message to its destination in O(logM) hops, where M is the number of nodes in the Chord overlay. In our system, M is much less than N, the total number of nodes in the system. Therefore, it should be very fast to locate an interest in the DHT overlay. Fig. 1 shows how a new node finds its interested community and joins to that community overlay. Note: a node can join several community overlays at the same time.

3.3. Query routing

Since peers with similar interests are in the same community overlay, we can expect that most queries would be satisfied within the local community. When a node initiates a query request, the request is propagated from that node to the whole community. Various strategies can be used for request forwarding inside the community overlay. For example, if controlled flooding such as the Gnutella searching protocol is used, the query is forwarded on to neighbors until the time-to-live (TTL) value reaches zero. We do not focus on how queries are routed within a community overlay, since it has been well studied in many literatures.

It is possible, though infrequent, nodes may want to search content in different categories. To resolve these query requests, communities which are in charge of the related interests have to be located first. Then queries can be forwarded to these communities and be answered by nodes there. The DHT overlay facilitates query forwarding among communities, just as it helps nodes join their interested communities. After the category of the query has been determined, the community in charge can be found by checking the category in the DHT overlay. Then the request is forwarded to entry nodes of the related community overlay, and is propagated in that overlay. To alleviate the lookup overhead on the DHT overlay, node will first check its own local host cache to see whether it has cached entry nodes falling into the target community. Moreover, it can also query neighboring nodes on their local host caches. Only after all these attempts fail, will the lookup request for entry nodes of a community be forwarded to DHT overlay network.

3.4. Overlay maintenance

It is necessary to maintain the DHT and community overlays to ensure that they remain working as nodes update their interests, join and leave the network. The DHT overlay uses its corresponding DHT membership protocol to maintain the overlay topology, for example Chord [18] or Pastry [17] protocol may be used. The load of DHT overlay comes from peers' register, update and lookup operations. To control the overhead, the update period can be adaptively adjusted. Both the register and update messages are lightweight: basically only the key of the interest and the IP address of the peer need to be recorded. Besides, only the most representative interests, usually only a few, are registered for each peer. To reduce the lookup overhead, nodes will check their own cache and their neighbors' caches first, before they lookup the DHT overlay network.

Nodes in the community overlay form an unstructured P2P network. The maintenance of the unstructured network like Gnutella is simple and lightweight. As mentioned, a new node joins the community overlay by connecting some entry nodes obtained from the DHT overlay. Inside the community overlay, nodes use Ping and Pong protocol to update neighbor table. In addition, nodes need to detect failures and repair faulty neighbors. The simplest approach to detect failures is to periodically send a hello message to every node in its neighbor table. Since its neighbors do the same, each node should receive a message from each neighbor in each period. If it does not receive this message, it probes the node and if the node does not reply it marks it faulty.

4. Experiment

We performed extensive simulations to evaluate the performance of the routing scheme. In this section, we first describe our simulation methodology, and then present results for different simulation scenarios.

The topology of the network defines the number of nodes and how they are connected. In our model, we used BRITE [11], a well-known topology generator to create two kinds of network topologies: the random graph and the power-law graph. The network is created by adding nodes one by one. And the performance test starts after all nodes have joined the network. The resource set includes 50,000 resources, and falling into 500 categories. We model the location of these resources using two distributions: the uniform distribution and a 70/30 biased distribution. Requesters are randomly chosen from the network. The dynamic network behaviors are simulated as this: in every unit simulation time, an active node has a 20% possibility to create a query, 1% possibility to update its resources, and 1% possibility to leave the system. Also, the same numbers of offline nodes join the system, and they start functioning without any prior knowledge. Our evaluation metrics are: (1) the recall rate which is defined as the number of results returned divided by the number of results actually available in the network; (2) the number of messages to forward queries; (3) the number of hops to resolve a query.

In the following experiments, we use Gnutella protocol as our intra-community searching protocol. To make comparisons we simulate our community-based searching scheme in conjunction with flat Gnutella searching.

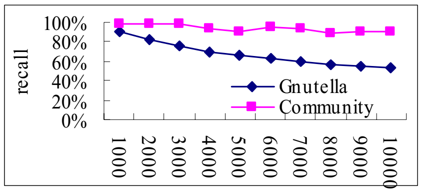

Figure 2. Recall rate versus network size

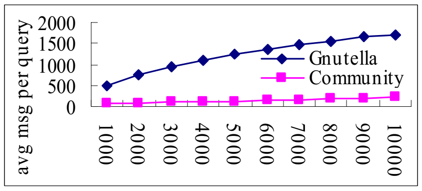

Figure 3. Message overhead versus network size

Fig. 2 and Fig. 3 compare the two routing strategies in terms of query recall rate and query message consumption. The network size increases from 1000 to 10000 and the TTL value is set 5. As expected, our community routing outperforms Gnutella routing on both metrics. In Fig. 2, Gnutella's recall rate decreases substantially as the network size increases, but our community routing is not directly affected by the network size. We can see that our routing scheme achieves a high recall rate even when the network size is very large. Fig 3 shows that our community routing sends significantly fewer messages than Gnutella does, to resolve a query, i.e., our system accrues lower costs. This is because, when given a request, the community routing can select a small number of overlay networks whose nodes have a higher number of hits. Nodes that have few results for this query will not receive it. Therefore, the searching space is reduced and queries get more results with certain TTL.

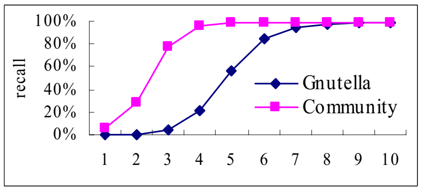

Figure 4. Recall rate versus TTL

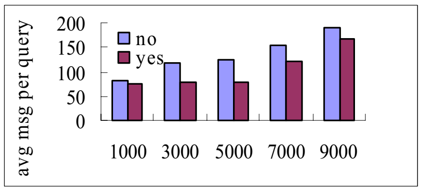

Figure 5. Effect of cache

Figure 4 illustrates the relationship of the query recall rate with the query TTL. The network size is 8000 in this experiment. Under the same TTL value, our community routing can achieve higher recall rate compared with Gnutella. We can see, in our system, a recall rate near 100 percent can be achieved with a very small TTL value. Figure 5 depicts the effect of using cache in the routing process. It is clear that using cache improves the searching performance by decreasing the message overhead.

5. Conclusion

As more and more resources appear in P2P networks, there is an increasing need to find an effective and efficient way to discover and query these resources. In this paper, we've presented an ontology-based community routing architecture to optimize search in P2P networks. This architecture integrates the advantage of both structured and unstructured P2P systems. It adopts a decentralized technique for identifying groups of nodes with common interests and build overlays that mirror shared interests. The community overlay searching scheme achieves higher efficiency and scalability than a pure flooding-based or history-based search scheme. At the same time it also retains desirable features of search in unstructured overlays such as self-organizing, fully distributed, scalable, and inherent support for rich resource descriptions and complex queries. Simulation experiments show that the proposed routing schemes are both efficient and scalable.

6. References

[1] Chawathe, Y., Ratnasam, S., Breslau, L. Lanhan, N. Shenker, S. "Making Gnutella-like P2P Systems Scalable", In *Proceedings of ACM SIGCOMM'03*.

[2] B.Yang, H.Garcia-Molina, "Efficient search in peer-to-peer networks", *Proc. of CDCS'02*, Vienna, Austria, July 2002

[3] Banaei-Kashani, F. and C. Shahabi. "Criticality-based analysis and design of unstructured peer-to-peer networks as complex systems". *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 351-358.

[4] Ajay Chander, Steven Dawson, etc., "NEVRLATE: Scalable Resource Discovery", *Proceedings of CCGRID'02.*

[5] T Berners-Lee, J Hendler, O Lassila, "The semantic web". *Scientific American*, 2001, 284(5):34–43.

[6] M. Schlosser, M. Sintek, S. Decker and W. Nejdl. "A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services". *Second International Conference on Peer-to-Peer Computing* (P2P'02) September 05-07, Linkoping, Sweden 2002.

[7] W. Nejdl, M. Wolpers, W. Siberski, A. Loser, I. Bruckhorst, M. Schlosser, and C. Schmitz. "Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-To-Peer Networks." *In Proceedings of the Twelfth International World Wide Web Conference* (WWW2003), Budapest, Hungary, May 2003.

[8] A. Halevy, Z. Ives, I. Tatarinov, and P. Mork. Piazza: Data management infrastructure for semantic web applications. In Proc. of the Int. WWW Conf., 2003.

[9] N. J.A. Harvey, M.B. Jones, S. Saroiu, M. Theimer, and A. Wolman. "SkipNet:A Scalable Overlay Network with Practical Locality Properties". *In Proceedings of the Fourth USENIX Symposium on Internet Technologies and Systems* (USITS '03), Mar. 2003.

[10] S. Shi, Y. Guanwen, D. Wang, J. Yu, S. Qu and M. Chen "Making Peer-to-Peer Keyword Searching Feasible Using Multi-level Partitioning". *Proc. Of the 3rd International Workshop on Peer-to-Peer Systems,* San Diego, CA, USA, February.

[11] A.Medina, A. Lakhina, I.Matta, and J. Byers, "BRITE: An Approach to Universal Topology Generation," Proc. The International Workshop on Modeling, *Analysis and Simulation of Computer and Telecommunications Systems- MASCOTS*, Cincinnati, Ohio, August 2001.

[12] B.Yang and H.Garcia-Molina, Designing a Super-Peer Ntrwork, *Proc. 19th Int'l Conf. Data Engineering*, IEEE Computer Society Press, Los Alamitos, CA, March 2003

[13] B.Yang and H.Garcia-Molina, "Designing a Super-Peer Ntrwork", *Proc. 19th Int'l Conf. Data Engineering,* Los Alamitos, CA, March 2003

[14] Lv, C., Cao, P., Cohen, E., Li, K., Shenker, S. "Search and replication in unstructured peer-to-peer networks". In: *ACM, SIGMETRICS* 2002.

[15] Gnutella website. http://gnutella.wego.com/

[16] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. "Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing," *Technical Report, UCB/CSD*-01-1141, April 2000.

[17] A. Rowstron and P. Druschel. "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms, Middleware*, November 2001.

[18] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H.Balakrishnan. "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *ACM SIGCOMM*, August 2001, pp. 149-160.

[19] S. Ratnasamy, P.Francis, M.Handley, R.Karp, and S. Shenker. "A Scalable Content-Addressable Network," *ACM SIGCOMM*, August 2001, pp. 161-172.

[20] Adamic, L., Huberman, B., Lukose, R., Puniyani, A.: "Search in power law networks". *Physical Review* (2001)

[21] P.Reynolds and A. Vahdat. "Efficient Peer-to-Peer Keyword Searching". *In Proceedings of ACM/IFIP/USENIX Middleware*, June 2003

[22] M. Cai, M. Frank, J. Chen and P. Szekely, " MAAN: A Multi-Attribute Addressable Network for Grid Information Services". *The 4th International Workshop on Grid Computing*, 2003.

[23] C.Tang and S.Dwarkadas. "Hybrid Gloablal-Local Indexing for Efficient Peer-to-Peer Information Retrieval". In *Proceedings of USENIX NSDI*, March 2004.

[24] Yahoo website. http://www.yahoo.com

[25] DMOZ website. http://www.dmoz.org

[26] Kazaa website. http://www.kazaa.com/