An Innovative Approach to Content Search Across P2P Inter-Networks

Potharaju S R P Saradhi **Mohmed Nazurudin Shaik** **Potharaju S R Aditya**

Abstract

Peer-to-Peer (P2P) computing has emerged as a popular model aiming at further utilizing Internet information and resources. Flooding is the basic method of searching in unstructured P2P networks; however, the blind flooding based search mechanism causes a large volume of unnecessary traffic, and greatly limits the performance of P2P systems. Our study shows that a large amount of this unwanted traffic is divinable and can be avoided while searching in P2P networks. In this paper, we aim at reducing the volume of unnecessary traffic, by proposing An Unnecessary Message Prediction based Searching mechanism (UMPS). UMPS is a pure distributed scheme, it is based on the distributed neighbor list where neighbors within two hops are stored in. UMPS cuts down unwanted traffic by terminating unnecessary flooding that has been predicted in advance, and the techniques related to distributed neighbor list are also discussed. Simulation results show that more than 60% unnecessary messages can be reduced by UMPS, and the query coverage range is retained at the same time, and more unwanted messages can be reduced if a peer has larger degree, therefore load balancing is achieved also.

Keywords: UMPS, P2P Network

1. Introduction

A distributed architecture consisting of a collection of resources (computing power, data, meta-data, and network bandwidth) performing a distributed function is called a **peer to peer architecture**. P2P computing is the sharing of computer resources and services by direct exchange between systems.

The huge popularity of recent peer-to-peer (P2P) file sharing systems has been mainly driven by the scalability of their architectures and the flexibility of their search facilities. Such systems are usually designed as unstructured P2P networks, because they impose few constraints on topology and data placement.

One of the most challenging problems in P2P research is the difficulty of locating content in an efficient and scalable way. Particular content is located by accessing the node(s) that manage content when the names or attributes of the desired content are specified.

In very large networks, it is not always easy to find desired resources. For any given system, the efficiency of any search technique depends on the needs of the application. Currently, there are two types of P2P lookup services widely used for decentralized P2P systems: structured searching mechanism and unstructured searching mechanism [4].

Structured systems such as Pastry are designed for applications running on well-organized

7th International CALIBER-2009,
Pondicherry University, Puducherry, February 25-27, 2009

networks, where availability and persistence can be guaranteed. In such systems, queries follow well-defined paths from a querying node to a destination node that holds the index entries pertaining to the query.

These systems are scalable and efficient, and they guarantee that content can be located within a bounded number of hops. To achieve this performance level, the systems have to control data placement and topology tightly within their networks. However, this results in several limitations: first, they require stringent care in data placement and the development of network topology. Thus, the tools they use are not applicable to the typical Internet environment, where users are widely distributed and come from non-cooperating organizations. Second, these systems can only support search-by-identifiers and lack the flexibility of keyword searching, a useful operation for finding content without knowing the exact name of the object sought. Third, these systems offer only file level sharing, and do not share particular data from within the files.

Unstructured systems like Gnutella are designed more specifically for the heterogeneous Internet environment, where the nodes' persistence and availability are not guaranteed. Under these conditions, it is impossible to control data placement and to maintain strict constraints on network topology, as structured applications require. Currently, these systems are widely deployed in real life. The advantages of unstructured systems over the structured systems drive us to concentrate on designing a new and efficient routing algorithm to locate content in unstructured networks. Hence, in our project, we concentrate on designing a new algorithm for converting any physical network into a conceptual network and implement a new query routing algorithm in the derived conceptual network to locate data present in the actual physical network.

2. Related Work

Peer to Peer (P2P) systems can take many forms. Email, Internet Relay Chat and Napster are all examples of P2P systems. Routing on these networks is either centralized or statically configured and is therefore unproblematic. Another class of P2P networks is the overlay network. Overlay networks build a virtual topology on top of the physical links of the network. Nodes leave and join this network dynamically and the average uptime of individual nodes is relatively low. The topology of an overlay network may change all the time. Once a route is established, there is no guarantee of the length of time that it will be valid. Routing in these networks is therefore very problematic and will be the focus of our report. Some of the issues facing designers of P2P routing algorithms are:

- Scalability
- Complexity

Scalability is a measure of how a system performs when the number of nodes and/or number of messages on the network grows. Complexity is the order of steps required for a packet to travel from one host to another in a worst case scenario.

3. Efficient Clustered Super-Peer

Efficient Clustered Super-Peer (ECSP) P2P model, aims at resolving the scalability and efficient query problems faced by unstructured P2P systems. ECSP follows a hierarchical approach; it allows pure P2P functions [9] independent of infrastructure, while

it provides advantages in scalability and search speed convergence.

3.1. Multi-tier Architecture

Peers in the system act as client peers and super-peers in different hierarchies. Super-peers act as local search hubs, building indices of the content files shared by each peer connected to them, and proxying search requests on behalf of these peers. Super-peers are connected with each other and organized amongst themselves into a backbone overlay network on the **super-peer tier.**

The hierarchical structure [10] of this system combines advantages of both centralized and pure P2P systems: it combines the efficiency of a centralized search with the autonomy, load balancing and robustness provided by distributed search mechanisms. Compared to centralized systems, the hierarchical structure distributes the load on the central server to many super-peers; therefore no single super-peer is required to handle a very large load, nor will one peer become a bottleneck or a point of failure for the entire system.

Compared to pure decentralized system like Gnutella, the hierarchical structure reduces the query traffic because only super-peers participate in searching and routing. Simultaneously, more nodes can be searched because each super node proxies for many regular nodes. Therefore, the introduction of a new level in the system hierarchy increases the scale and speed of query lookup and forwarding processes. Moreover, the hierarchical structure is more stable because clusters join and leave the network less frequently than individual peers.

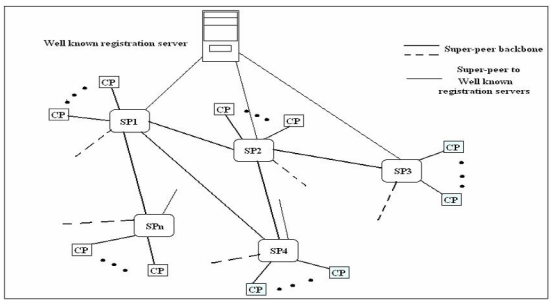

Figure 2.1 Clustered Super-peer architecture

3.2. System Modules

3.2.1.Well-known Server Module

Registration servers maintain databases of all active super-peers in the system, and when a new super-peer is added to the network, a new entry is generated in the registration

server's super-peer database. The registration server then sends a neighbour list to the super-peer, which includes a set of super-peers already in the system, and the new super-peer can join the network by connecting to these neighbours. Neighbours in the neighbour list are not chosen randomly, but rather they are the nodes that are topologically closest to the new neighbour. When a new client peer joins the system, it first contacts the registration server to get a super-peer list and to identify the super-peers located closest to it topologically.

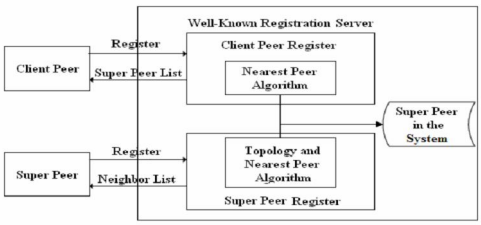

Figure 2.2.1 Well-known registration server structure

3.2.2. Super-peer Module

Super-peers are selected from regular peers according to the super-peer selection algorithm that

we propose based on the proximity criteria. The super-peer selection algorithm is explained in section 3. Super-peers act as cluster leaders and service providers for a subset of client peers, providing four basic services to the clients: join, update, leave and query.

After obtaining a super-peer list from a well-known registration server, client peers choose one super-peer from the list and connect to it. In the join process, client peers upload metadata describing the property of the content they will share with the network.

When a client peer leaves the system, the super-peer removes that client peer's metadata from the index library. If a client peer ever updates its content data, it sends an update message to the super-peer, and the super-peer updates its index accordingly. When a super-peer receives a query from its client peer, it matches what is in its index library and forwards the query to its neighbours, who in turn forward it to some of their neighbours.

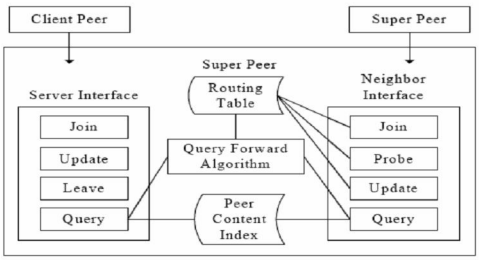

Fig ure 2.2.2 Super-peer structure

3.2.3. Client-peer Module

Regular peers are referred to as client peers to distinguish them from super-peers. In fact, they act as both clients and servers: they send requests to super-peers like clients, and receive other peers' file download requests like servers. After the client peer joins the system and uploads its content metadata to its local super-peer, it initiates an FTP server on a well-known port and waits for other peers' download requests. After a client peer locates content through super-peers, it opens a connection and downloads directly from the node where the content is located.

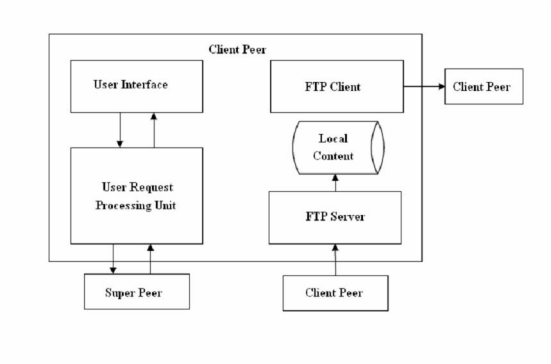

Figure 2.2.3 Client-peer structure

3.2.4. Backup-peer Module

The introduction of one more level of hierarchy makes the system more efficient, but the super-peer becomes a potential area of single-point failure for its cluster. To increase the reliability of the system, we introduce a backup peer as redundancy for the super-peer. Thus, every cluster has a super-peer acting as a cluster leader and a backup peer [11] acting as a redundancy server. The backup peers are selected from the client peers too using the same super-peer selection algorithm explained in section 3. They copy the super-peer's index table periodically, and when a super-peer fails or leaves the network, its backup peer replaces it and the cluster selects a new backup peer for redundancy.

3.3. Distance Measurement Strategy

The distance between peers in the underlying overlay network can be represented using latency of a packet delay or hop count between them. Traceroute can trace the route that a packet traverses to host in the network by launching UDP probe packets. We can get the RTT (RoundTrip-Time)

value and the TTL (Time-To-Live) value from a returned message by running the traceroute instruction on a client. The RTT value denotes latency, and the TTL value denotes hop count between two hosts.

Because the RTT directly reflects the packet delivery latency between two peers, RTT can be used to indicate the topological distance between two peers. However, RTT could fluctuate even for the same route due to network traffic change within a short period of time. To accurately find the closest super-peer to a peer, TTL can provide an auxiliary metric.

Let **dist(p,q)** represent the underlying distance between peers **p** and **q**. Let $\mathbf{dist_{RTT}(p,q)}$ to represent the detected RTT value and $\mathbf{dist_{TTL}(p,q)}$ to represent the detected TTL value. Let **d** be a predefined value to denote the latency difference of two paths. Given peers **p, q, s, t** then the distance can be compared using the following concept:

If $|\mathbf{dist_{RTT}(p,q) - dist_{RTT}(s,t)}| > \mathbf{d}$ **then**

  If $\mathbf{dist_{RTT}(p,q) \le dist_{RTT}(s,t)}$ **then**

    dist(p,q)> dist(s,t)

  Else dist(p,q)< dist(s,t)

Else If $\mathbf{dist_{TTL}(p,q) \le dist_{TTL}(s,t)}$ **then**

  dist(p,q)> dist(s,t)

  Else dist(p,q)< dist(s,t)

3.4. Client-peer Joining Algorithm

Let $\mathbf{H\{p_0,p_1,\ldots,p_{k-1}\}}$ be a set of peers in the network, where **k** is the number of peers in the network. Let $\mathbf{P\{sp_0,sp_1,\ldots,sp_{m-1}\}}$ where **m** is the number of super peers in the network.Let $\mathbf{SP_i\{sp_i|b_0,b_1,\ldots,b_n\}}$ **(i=0,1,…m-1**) be the **n** peers under the super-peer $\mathbf{sp_i}$.

Step 1. The new client peer **u** measures the distance from itself to every super-peer in **P** in terms of RTT value $\mathbf{dist_{RTT}(u,sp_i)}$ and the TTL value $\mathbf{dist_{TTL}(u,sp_i)}$ **(i=0,1,…,m-1).**

Step 2. Find a super-peer $\mathbf{sp_j}$ from P, such that $\mathbf{sp_j} \in$ P and $\forall$**q** P, **dist(u** $\mathbf{sp_j}$**,) d" dist(u,q)**, i.e. the super-peer $\mathbf{sp_j}$ is the closest super-peer to the peer **u**.

Step 3. The client peer joins the cluster that contains the super-peer $\mathbf{sp_j}$ and $\mathbf{SP_j = SP_j \cup \{u\}}$ where 0d" jd" m-1 .

3.5. Neighbour Information Based Flooding Algorithm

A decentralized unstructured P2P network considered is defined as a graph G, there are N peers which are typically user-machines in G, and there is not more than one edge

may connect a specific pair of peers. To describe our algorithm more clearly, other terminologies are defined as follows:

*A_i: The identifier of peer i.

*hop_t: the network distance that is t hops distance from the specific central peer.

*Nbr(A_i, hop_t): The neighbour set of Ai, and the shortest distance between peers in the set and A_i is exact t hops.

*Unnecessary message: The repeated flooding query message received by a peer.

*Effective message: Flooding query message except for unnecessary messages is also named necessary message.

*Msg(s, d, type, content): Message transferred among peers.

S is the source peer, **d** is the destination peer, and **type** is the type of the message, **content** is the content actually to be transferred. The type of flooding query message is 0, and query message that is sent from A_i to A_j is simply described as $Msg(A_i,A_j)$ and A_i is called father of A_j.

3.5.1. Rules for Predicting Unnecessary Messages

Two conditions must be provided for a peer to predict unnecessary messages. One is that the peer is able to know its father, in a flooding case, a peer's father may be gotten from the source address in the received message packet; The other is that the peer knows the topology information of neighbours within two hops, this is attained by utilizing the neighbors' direct neighbours, how to create and maintain the neighbour topology information will be amply represented later.

Assumption: In a P2P system, A_i is father of A_j, that is to say A_j has received message from A_i, and A_j will send the message to its neighbours (A_k) except A_i, so A_k Nbr(Aj, hop1) and A_k '" A_i, A_j has neighbour topology information within two hops. Then A_j can predict that other peers may also send the messages to Ak, define these peers (including A_j) as Possible Father Set (PFS) of Ak, expressed as $PFS(A_i, A_j, A_k)=\{Nbr(A_k, hop_1)\ Nbr(A_i, hop_1)\}$.

Then, **Unnecessary Message Prediction Rule (UMP)** is described as follows:

Rule 1: if A_k $Nbr(A_i, hop_1)$, then $Msg(A_j, A_k)$ must be unnecessary message.

Rule 2: if A_k $Nbr(A_i, hop_1)$, and $PFS(A_i, A_j, A_k)$ '" $\{A_j\}$, then $Msg(A_j, A_k)$ is possible unnecessary message, or else it is necessary message.

If the each peer in P2P system can detect and stop the unnecessary messages the total network traffic could be significantly reduced without shrinking the search scope of queries. So in UMP, by utilizing direct neighbours, a Neighbour Information Table (NIT) is constructed on each peer, before a peer is to flood a query, the validity of the message is predicted by using UMP rule according to NIT, if the query message is necessary, then send it, or else stop it. This is the basic principle of UMP. Two operations are defined in UMP. The first operation is creating and maintaining the NIT, including peer adding, peer leaving and NIT information updating. The second operation is flooding query with unnecessary query message predicting.

3.5.2. Proposed Algorithm

NIT is employed by each peer to store the neighbour topology information, the content in a NIT consists of two parts: the central peer's all direct neighbours; and these direct neighbors' direct neighbours correspondingly, i.e. peers that are two hops distance from the central peer.

The hop2 neighbours may attain from hop1 neighbours by sending specific request messages. The topology within two hops can be gotten according to NIT easily. But the topology of P2P network varies from moment to moment, therefore constructing NITs for new peers and maintaining NITs for resident peers are necessary. Operations related to NIT are peer joining, peer leaving and updating. Four types messages related to NIT operations are used:

*Msg(s, d, 1): s asks d for hop1 neighbours of d.

*Msg(s, d, 2): s sends hop1 neighbours to d.

*Msg(s, d, 3): If s is a new peer, s will send this message to its direct neighbours.

*Msg(s, d, 4): If s will leave the network, this message will be sent to direct neighbours of s.

Algorithm 1: Update NIT

Step1: A_i sends Msg(A_i, A_j, 1) to A_j, if A_jNbr(A_i, hop_1).

Step2:If A_j receives Msg(A_i, A_j, 1), then reply Msg(A_j, A_i, 2) to Ai. The content in reply packet is mutative part of A_j's direct neighbours that extracted from its NIT within a cycle, if there is no change, and then fills the content part with "NoChange" flag that is defined in advance.

Step3: If A_i attains a reply, then checks the packet, if the content is "NoChange", does nothing, or else, updates its NIT according to the content in the packet.

Algorithm 2: Peer Joining

Step1: A_i sends Msg(A_i, A_j, 3) to A_j, if A_j Nbr(A_i, hop1), content of the packet includes Ai's direct neighbours.

Step2: If A_j receives Msg(A_i, A_j, 3), then reply Msg(A_j, A_i, 2) to A_i. The content in reply packet is direct neighbours of A_j, and then, updates its NIT, at last, sends Msg(A_j, A_k, 2) to its direct neighbours (A_k) except A_i.

Step3: If A_i attains a reply, then constructs (updates) its NIT.

Step4: If A_k attains a message from A_j, then updates its NIT.

Algorithm 3: Peer Leaving

Step1: A_i sends Msg(A_i, A_j, 4) to A_j, if A_j Nbr(A_i, hop_1), then leave the system.

Step2: If A_j receives Msg(A_i, A_j, 4), then updates its NIT, and sends Msg(A_j, A_k, 2) to Ak. If A_k Nbr(A_j, hop_1), The content in the packet is a flag, the flag indicates that Ai has left network.

Step3: If A_k gets message from A_j, then updates its NIT.

Algorithm 4: Search Flooding

Step1: If A_i has received the query before, drop the query.

Step2: If the query is satisfied, then return results to sponsor.

Step3: If TTL is not more than zero, drop the query.

Step4: For each A_j Nbr(A_i, hop_1), predicts the validity of Msg(A_i, A_j) by UMP. If it is a necessary message, then send it to A_j; if it is unnecessary message; do not send it; if it is a "possible" unnecessary message judged by UMP Rule2 then do not send it.

4. Experimental Results

The graph comparing all the three proposed algorithms has been generated by taking into account the '.tr' trace files created due to the generation of various networks. The graph has been plotted between 'number of nodes in the network' on x-axis and the 'packets dropped' in each network corresponding to the respective algorithm, on y-axis.

The "red" line represents "general flooding algorithm", the "green" line represents "restricted path flooding algorithm" and the "blue" line represents the "NIT based flooding algorithm".

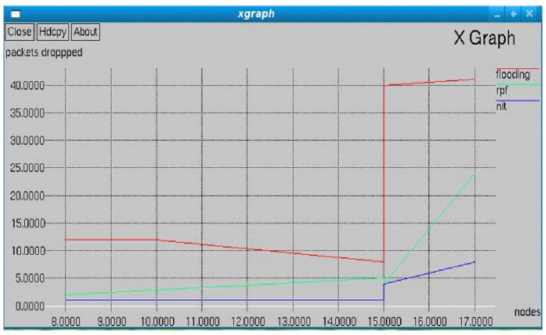

Fig ure 5.5.1 Number of Nodes vs Number of Unnecessary Messages

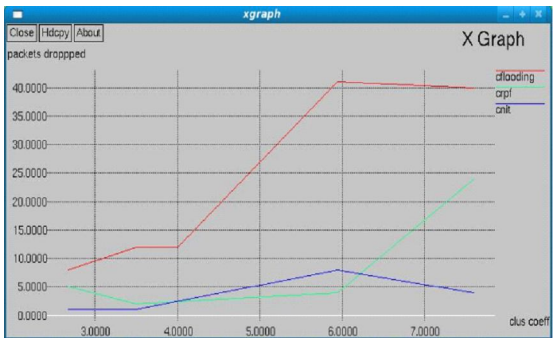

Figure 5.5.2 Clustering Coefficient vs Number of Unnecessary Messages

Our simulation results show that:

- In the "general flooding algorithm", 70% of the total messages generated are unnecessary.
- In the proposed "restricted path flooding algorithm", 42% of the total messages generated are unnecessary.
- In the proposed "Neighbour Information Table based flooding", algorithm, 27% of the total messages generated are unnecessary.

References

1. **Vinod Muthusamy,**"An Introduction to Peer-to-Peer networks", Presentation for MIE456 – Information Systems Infrastructure II, 2003.

2. **Jem E. Berkes,** "Decentralized Peer-to-Peer Network Architecture: Gnutella and Freenet", University of Manitoba, Canada.

3. **Song Jiang, Lei Guo, Xiaodong Zhang**, "Light Flood: an Efficient Flooding Scheme for File Search in Unstructured Peer-to-Peer Systems", Proceedings of International Conference on Parallel Processing, IEEE 2003.

4. **Mayank Bawa, Brian F.** Cooper, Arturo Crespo, "Peer-to-Peer Research", www.sigmod.org/sigmod/record/issues/0309/B5.awa.pdf

5. **Peter Backx, Tim Wauters, Bart Dhoedt, Piet Demeester,** "A comparison of peer-to-peer architectures".

6. **Qin Lv, Pei Cao, Edith Cohen, Kai Li, Scott Shenker,** "Search and Replication in Unstructured Peer-to-Peer Networks".

7. **Christos Gkantsidis, Milena Mihail, Amin Saberi,** "Random Walks in Peer-to-Peer Networks", IEEE 2004.

8. **Shudong Jin, Hongbo Jiang,** "Novel approaches to efficient flooding search in peer-to-peer networks", ACM journal, Dec 2006.

9. **Vicent Cholvi, Pascal Felber, Ernst Biersack,** "Efficient Search in Unstructured Peer-to-Peer Networks".

10. **Ching-Hsien Hsu, Chih-Hsun Chou, Chi-Guey Hsu and Shih-Chang Chen,** "On Improving Message Passing in Unstructured Peer-to-Peer Overlay networks", The 3rd International Conference on Grid and Pervasive Computing, 2008 IEEE.

11. **William Acosta, Suresh Chandra,** "Improving Search Using a Fault-Tolerant Overlay in Unstructured P2P Systems",

International Conference on Parallel Processing, 2007 IEEE.

12. **Ken Y. K. Hui, John C. S. Lui, David K. Y. Yau,** "Small world overlay P2P networks", IEEE2004.

13. **Shashidhar Merugu, Sridhar Srinivasan, Ellen Zegura,** "Adding structure to unstructured peer-to-peer networks: the use of small-world graphs", 2004 Elsevier.

14. **Sabato Manfredi, M. di Bernardo, Franco Garofalo,** "Small world effects in networks: An engineering interpretation", 2004 IEEE.

15. **Tsungnan Lin, Hsinping Wang,** "Search Performance Analysis in Peer-to-Peer Networks", Proceedings of the Third International Conference on Peer-to-Peer Computing, IEEE 2003.

About Authors

Mr. Potharaju S R P Saradhi, M.S -Software Engineering
E-mail: psrpardhasaradhi@yahoo.co.in
Mr. Mohmed Nazurudin Shaik, M.S -Software Engineering
E-mail: nazuruddin@yahoo.com
Mr. Potharaju S R Aditya, M.S -Software Engineering
Vellore Institute of Technology Vellore Institute of Technology
E-mail: aditya992005@gmail.com