Journal of Theoretical and Applied Information Technology

© 2005 - 2009 JATIT. All rights reserved.

www.jatit.org

P2P INFORMATION RETRIEVAL FRAMEWORK FOR DIGITAL LIBRARY SYSTEM

[1]A.R.RENUGA, [2]B.DR.G.SUDHASADASIVAM
[1]Research Scholar,PSG College of Technology, Coimbatore,Tamilnadu, India

[2] Asst. Professor, PSG College of Technology, Coimbatore,Tamilnadu, India.
E-mail : renugacit@yahoo.co.in, sudhasadasivam@yahoo.com

ABSTRACT

P2P Information Retrieval Framework is a modification of the pastry framework for distributed search application**.** This framework consists of a peer to peer network of nodes, which voluntarily agree to share their resources by joining the network. While joining the network these nodes construct the active peer list. The files are distributed over the peer to peer network based on the keywords. The searching request could be initialized in anyone of the peers in the peer to peer network. The search request consists of keywords from which unwanted words are removed. This search request is propagated in an incremental fashion across the nodes with the aim of finding the best node in the incremental fashion. The results obtained from each keyword are aggregated and the final result is listed in the node which initiated the search procedure.

Key Words - *Pastry, SHA, Hashing, Distributed Hash Table.*

1. INTRODUCTION

A peer to peer network is a network in which all the nodes have equal priority. A peer-to-peer (P2P) computer network exploits diverse connectivity between participants in a network and the cumulative bandwidth of network participants rather than conventional centralized resources where a relatively low number of servers provide the core value to a service or application. The network bandwidth in a P2P network is fully utilized. As nodes in the network are interconnected, there is no single point of failure. P2P networks are characterized by high processing power and storage without the overhead of high cost hardware. P2P networking has the potential to greatly expand the usefulness of the network be it for sharing music and video, privately contracting for services or for coordinating the use of expensive scientific instruments and computers.

Peer-to-peer networks have become popular of late for a range of applications, including the well known file sharing systems such as Gnutella and Bit Torrent. Peer-to-peer in general refers to distributed systems where resources are obtained from an ad hoc collection of client computers. Typically the set of peer nodes participating in the network changes over time. Early P2P systems such as Napster used underlying client-server architecture to facilitate communication between peers. Later, fully decentralized (pure) P2P networks where developed to eliminate the impact of a single point of failure and to make it cheaper to scale. These early pure P2P networks were inefficient, requiring a great deal of network bandwidth which prevented their use on lower bandwidth connections. Their design also meant that data present in “distant” parts of the network may not always be found. Pure P2P research has since centered on the idea of a distributed hash table (DHT). A hash function is used to map objects to be stored in the P2P network to the peer node that should hold that object. The peer node whose randomly assigned identifier is “closest” to an object’s hash value will always hold that data. As the set of peers that make up the P2P network changes over time, the ownership of data changes so as to maintain the above invariant. It is this invariant that allows messages to be efficiently routed (average O (log N) hops) to arbitrary objects without needing to update object references and without needing to resort to forwarding schemes. Distributed Hash Tables are systems that allow key based insertion, lookup and deletion of objects in a distributed setting [1-4].

Such peer-to-peer DHTs form the basis of several applications such as file sharing, storage, multiplayer games etc. [5-9].

Pastry [10] is one such DHT that has been adapted for several different applications [6-7]. Pastry networks consist of a set of machines (nodes) which are assigned a unique k-bit key termed as nodeID. The assignment of nodeID to nodes must be done in a fully decentralized manner but in such a way that the nodeIDs are approximately evenly distributed across the address space. Nodes maintain information to facilitate the routing of messages. Each node remembers a small set of nodes which are physically closest termed as neighborhood set nodes and a small set of nodes which are virtually closest termed as leaf set. This information is stored in a routing table.

When presented with a message and a key, a Pastry node efficiently routes the message to the node with a Node ID that is numerically closest to the key, among all currently live Pastry nodes, which can be visualized as a ring. Each Pastry node keeps track of its immediate neighbors in the Node ID space, and notifies applications of new node arrivals, node failures and recoveries. Pastry takes into account network locality and is completely decentralized, scalable, and self-organizing; it automatically adapts to the arrival, departure and failure of nodes. The expected number of routing steps is O(log N), where N is the number of Pastry nodes in the network. Peer to Peer cycle stealing framework [11-13] can be used to efficiently host a distributed application.

The proposed framework is a modification of Pastry and is specifically created to host the digital library application that searches efficiently for files in the network. This framework makes use of the properties of pastry with a slight modification to suit the digital library application. In order to make the peers aware of its neighbors this proposed approach creates an active peer list. Routing table and the leaf set is then created from the active peer list. Neighborhood set is not used in this framework. This approach uses remote method invocation to forward the search request and to start the searching process in other nodes.

2. FRAMEWORK

The proposed framework consists of a ring formed by the set of peers which have agreed to take part in the searching process. Each peer maintains information about the set of peers which are closest to it. This information is found in the leaf set stored in routing table. The pictorial representation of proposed framework is shown in Figure 1.

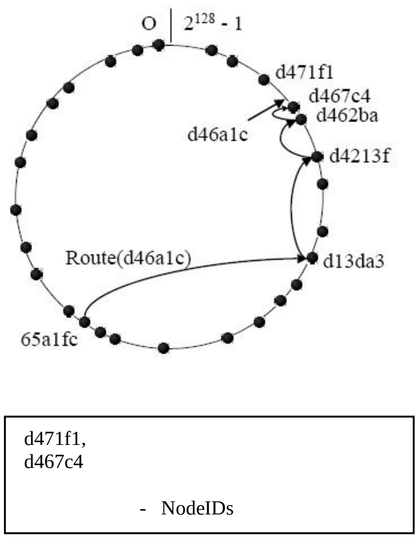


Figure 1. Proposed Framework

The routing table contains information about the set of peers whose nodeIDs have the prefixes that match with the present nodeID. The leaf set maintains information regarding the set of peers having nodeIDs which are closest to the present nodeID in terms of the value of their hashed IP address.

3. SYSTEM DESIGN FOR IMPROVING COMMUNICATION LOCALITY

3.1. Node Joining

Node Joining is the initial process carried out before commencing the search procedure. This is the process of acquiring all the active peers that are ready to donate their resources. There are no constraints regarding the number of nodes that should participate in the searching process. Once the database is distributed, there cannot be further addition of peers into the network. During the process of node joining the peers broadcast a single packet to all the listening peers. After broadcasting for a particular period of time, the peers stop sending packets and the peers list is constructed in all the active peers. The peers list contains the IP address of all the active peers. From this information a peer can construct its routing table and the leaf set. This information is used by the peer for the routing purpose.

3.2. NodeID Assignment using Hashing

Hashing is done for the IP addresses and the keywords. Secured Hashing Algorithm (SHA-1) is used to produce the 160-bit hash value. This is converted into 80 digit value for the convenience of constructing the routing table. Figure.2 explains NodeID assignment process.

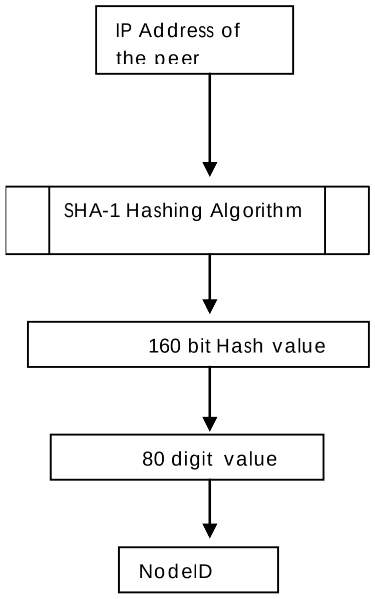


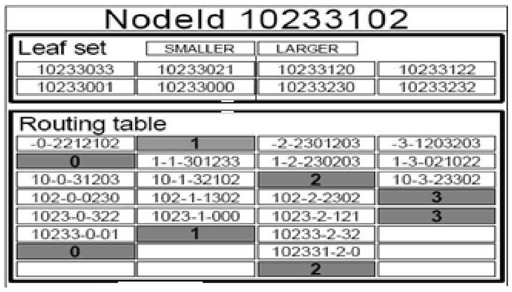
Figure 2. IP Address translate into NodeID

3.3 Routing Table Construction

The IP addresses of the active peers are hashed. This hash value gives the nodeID of the peer node. The routing table is then constructed using the hash value. Prefix match is then found between the participating nodeID and the present nodeID (i.e., the node for which the routing table is being constructed now). The number of prefix digits matched forms the row in the routing table. The value of the first unmatched digit forms the column in the routing table. All the other entries are left empty. The routing table entry which corresponds to the present nodeID is also left empty. Figure 3. Shows routing table.

3.4. Leaf Set Construction

Eight closest nodeIDs to the present nodeID form the leaf set of which four entries are less than the present nodeID and the remaining four entries are greater then the present nodeID.



NodeId 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Routing table

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Figure 3. Routing Table of the proposed framework

4. SYSTEM DESIGN FOR DIGITAL LIBRARY SYSTEM

4.1. Database Distribution

The application is made suitable for distributing the files of all the active peers. This is done by requesting the user for the set of most appropriate keywords for a file to be distributed across the network. The received keywords are hashed using the same hashing function used for hashing the IP addresses of the peers. Hence the hash value for a keyword is also an 80 digit value. This hash value is compared to the IP hash values of the closest peers and the peer having the greatest prefix match to the keyword hash value is selected for distribution. Hence the keyword is sent to that peer and there again the same process of finding the peer with greatest prefix match is done. If the present peer is the best peer for that keyword then the file is stored in that peer. This process is repeated for all the set of keywords available for the file.

4.2. Searching

Searching is done for the search request received on a peer. The received request is converted into set of keywords by removing unwanted words from the search request. These sets of keywords are again hashed to obtain the 80 digit value. After hashing the keywords, peer to peer searching is done for obtaining the peer which may contain files that match the search keyword. After finding the peer where the files for that search keyword might be found, the database of that peer is searched. This searching is called as database searching. This process is repeated for all the keywords found in the search request. The resulting answer is sent to the peer who initially received the searching request.

4. 3. Populating the database

To add a file to the database the file is need to be uploaded to the network. Uploading the file involves keyword selection. The user uploading the file should provide the appropriate keywords for the file. These keywords should be selected in an appropriate manner to provide the file when search is made. The keywords are hashed to produce the 80 digit key which is similar to the format of the nodeID. The peer who uploads the file searches for a closer nodeID to the hash key than itself. This process continues in all the peers to which the upload request is forwarded till there is no closer nodeID than the present nodeID. After the nodeID has been found the file is transferred to that peer from the peer who initiated the upload request. Similarly this process is repeated for all the keywords for that file. Hence the file may be placed in more than one peer due to different keywords.

4.4.Peer to peer searching

Peer to peer searching [14] is the upper level search done to find the peer where the file required is located. The given keyword that has to be searched is hashed. Using the routing table of the present node, the nodeID which is closer to the keyword hash than the present nodeID is located. Whenever a request reaches a peer, the following steps are taken

1) Search for the match of keyword hash with NodeID (row no)
2) Identify next unmatched digit in the keyword hash as column.
3) Forward the search request when entry is there.

5. IMPLEMANTATION DETAILS

This application is implemented on systems with Processor: Intel® Core™2 Duo, Frequency 1.86 GHz y RAM 1015 MB, 32-bit Operating System (Windows Vista™ Ultimate).

First task in implementation is to setup the network for the digital library application. The setting up of the network consists of creation of the active peer list. This list is created by means of broadcasting. When a new node wants to join the network and become a peer then it starts off by broadcasting a packet in the network. This broadcasting is done until other peers which want to take part in the search request have also started their broadcasting. When all the peers have started broadcasting then can be stopped the broadcasting. At the end of this process, a file containing IP addresses of all the peers in sorted order is constructed.

5.1. Uploading Module

Once the network is setup, the database should be distributed. The appropriate keywords are given as input to the application. Each keyword is then hashed using SHA to obtain 80 digit hash value.

The keyword hash value is checked with the leaf set to see if it lies within the range of the leaf set. If it lies within the range then the request is sent to the peer having the closest match to the keyword hash. If it does not lie inside the leaf set then the prefix match of the keyword hash value and the present IP hash value is found. Then the next better nodeID with a higher prefix match is found from the routing table. If the routing table entry is empty then we take the best peer from all the active peers available. This process is incrementally repeated until no peer has a better nodeID than the present nodeID.

5.2. Searching module

Once the database has been distributed the searching process is initiated by the user. Search request consists of a string which is divided into search keyword list and unwanted words list. This segregation is done by utilizing a dictionary with the unwanted words. The words obtained from the search request string are matched with the entries of the dictionary. If any match occurs then that particular word is removed from the search request. This dictionary is maintained in all the active peers. Once the keywords have been identified from the search request the searching process is initiated.

The keywords are again hashed to obtain the 80 digit hash value. This search keyword hash value is checked with the leaf set to see if it lies within the range of the leaf set. If it lies within the range then the request is sent to the peer having the closest match to the search keyword hash. If it does not lie inside the leaf set then the prefix match of the search keyword hash value and the present IP hash value is found. Then the next better nodeID with a higher prefix match is found from the routing table. If the routing table entry is empty then the best peer from all the active peers available is taken. This process is incrementally repeated until no peer has a better nodeID than the present nodeID. Once the peer on which the files for the required keyword is found the database on the peer searched. The default directory is searched for the list of files where the keyword has occurred at least once. This result is sent back to the peer who initially received the search request.

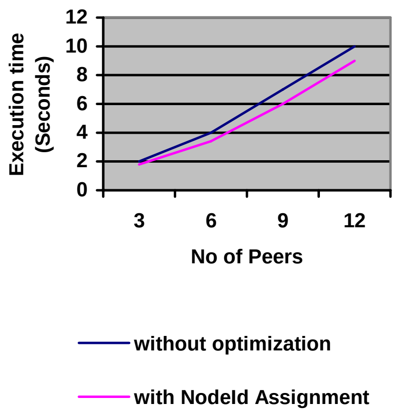


Figure.4. Performance Analysis (No of peers vs. Execution Time)

Figure.4. Describes the performance analysis of the proposed framework.

6. CONCLUSION

The digital library application takes advantage of the features given by peer to peer network and the pastry framework. It searches for a particular keyword in a group of nodes connected to the network. The group of nodes is evenly distributedacross the address space and hence the searching process becomes more balanced among the individual peer. As the network bandwidth is efficiently used the searching time is reduced.

ACKNOWLEDGEMENT

The authors would like to thank Dr.R.Prabhakar, the Principal, Coimbatore Institute of Technology, Dr.RudraMoorthy, Principal, PSG College of Technology and Mr. Chidambaram Kollengodu, YAHOO Software Development (India) Ltd, Bangalore for providing us the required facilities to complete the project successfully. This project is carried out as a consequence of the YAHOO's University Relation Programme.

7. REFERENCES

[1] The Gnutella protocol specification v 0.4, Document revision 1.2, www.clip2.com.

[2] I. Gupta, K. Birman, P. Linga, A. Demers and R. van Renesse. Kelips(2003):Building An efficient and stable P2P DHT through increased memory and background overhead, In Proceedings of International Workshop on Peer-to-Peer Systems , pp. 81-86.

[3] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek and Balakrishnan(2001):Chord: a scalable peer-to-peer lookup service for Internet applications, SIGCOMM 2001, pp.149-160.

[4] A. Rowstron and P. Druschel(2001):Pastry: scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer SystemsIFIP/ACM International Conference on Distributed Systems Platforms , pp. 329

[5] B. Knutsson, H. Lu, W. Xu and B. Hopkins(2004):Peer-to-Peer Support for Massively Multiplayer Games, The Annual Joint Conference of the IEEE Computer and Communications Societies .

[6] A. Rowstron and P. Druschel(2001):Storage management and caching in PAST, a largescale, persistent peer-to-peer storage utility, In Proceedings of Symposium on Operating Systems Principles. pp. 199-201.

[7] J. Chase, B. Chun, Y. Fu, S. Schwab, and A. Vahdat. Sharp(2003): An architecture for secure resource peering, In Proceedings of Symposium on Operating Systems Principles . pp. 170- 184.

[8] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R.Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao(2000):OceanStore -An Architecture

for Global-Scale Persistent Storag, In Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems .

[9] I. Clarke, O. Sandberg, B. Wiley and T. Hong(2002):Freenet: A Distributed Anonymous Information Storage and Retrieval System, In Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability.

[10] Antony Rowstron & Peter Druschel(2001): Pastry: Scalable, ecentralized object location and routing for large- Scale peer-to-peer systems,in Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms

[11] Richard Mason & Wayne Kelly(2007): Enhancing Data Locality in a Fully Decentralized P2P Cycle Stealing Framework in Thirtieth Australasian Computer Science Conference.

[12] Mason, R. & Kelly, W.(2005):G2-p2p: A fully decentralized fault- tolerant cycle-stealing framework, in 'Proceedings of the Australasian Workshop on Grid Computing and e-Research (AusGrid05),Newcastle, Australia.

[13] Mason, R. & Kelly, W:Peer-to-peer cycle sharingv ia .net remoting, in Proceedings of the Ninth Australian World Wide Web Conference (AusWeb03)'.

[14] Xiuqi Li & Jie Wu, "Searching Techniques in Peer-to-Peer Networks".