International Journal of Computer Applications (0975 – 8887)
Volume 50 – No.4, July 2012

Tag-based Improved Search in Peer-to-Peer Overlays

Thummala Sreeja
BITS Pilani Hyderabad Campus
Jawahar Nagar, Shameerpet
Hyderabad, India, 500078

Chittaranjan Hota
BITS Pilani Hyderabad Campus
Jawahar Nagar, Shameerpet
Hyderabad, India, 500078

Antti Ylä-Jääski
Aalto University, Helsinki
P.O. Box: 11000, Espoo
Helsinki, Finland

ABSTRACT

P2P based file sharing applications have gained immense popularity in recent years. Most content sharing applications search by matching the query with the file name. Some content sharing applications have gone a step further and used metadata assigned to it by the user who shared the content for searching the content. This might not be the best approach as the tags assigned by the user may be biased and inadequate in describing the content. In this paper, an improvised search is proposed in distributed P2P content sharing systems which make use of the external metadata assigned to the content by the public. The search is implemented on JXTA overlay for its interoperability and tested. Results of a comparative study that involves the improvised tag based search are included. In addition to this, a mathematical argument is provided in support of the approach.

General Terms

Peer to Peer; Collective Tagging; File Sharing; Juxtapose.

Keywords

Tags; Search; JXTA; P2P; Stemming; Overlay.

1. INTRODUCTION

Client-server based architectures are characterized by asymmetric relationship between client and server where client queries and server responds. Contrast to that in distributed P2P systems every node acts as both a server and a client. Due to the increased bandwidth, high storage capacity P2P systems are increasingly becoming another way to create web integrated applications [1]. Peer-to-Peer overlay structure is brought to popularity by the file sharing applications like Napster [2], Gnutella [3] etc. A P2P overlay network is a logical network at application layer providing connectivity, routing, and messaging amongst addressable end-points of the communication. These overlay networks have been used for file sharing, Voice over P2P, and streaming media over P2P in recent times. The spring 2011 global Internet phenomena report by Sandvine points to the increased percentage of P2P traffic in North American fixed access networks from 15.1% in 2009 to 18.8% in 2011 as shown in Fig.1. Most content sharing applications search by matching the query with the file name. Some content sharing applications have gone a step further and used metadata assigned to it by the user who shared the content for searching an object. Further, these objects are annotated and rated by different peers. These tags and derived peer specific scores can be leveraged for algorithm called SLAC, which is based on tags for the purpose of cooperation amongst the peers in a P2P network. searching relevant content and discovering subjectively important and interesting items or objects in a P2P network. This might not be a best approach as the tags assigned by the user may be biased and inadequate in describing the content. In this paper, an improvised search is proposed in distributed P2P content sharing systems which make use of the external metadata assigned to the content by the public. The search is implemented on JXTA overlay for its interoperability.

Section 2 describes related work. A brief introduction of JXTA is given in Section 3. The details of architectural design of the search and its implementation are given in Section 4. Results of a comparative study that involves the improvised tag-based search are included in Section 5. A mathematical argument that supports the proposed approach is provided in Section 6. Finally, Section 7 concludes the work and proposes the future extensions.

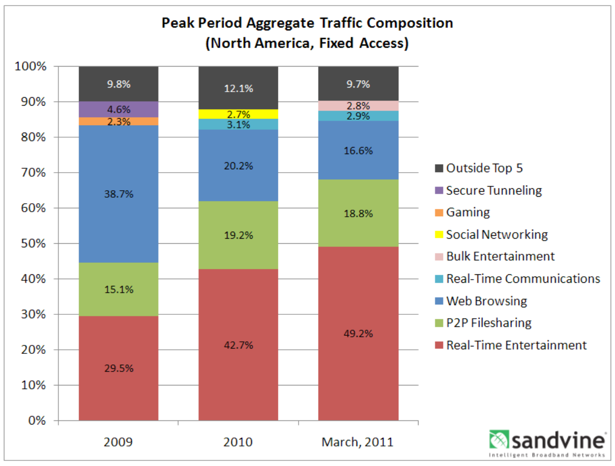


Fig 1: Aggregate traffic in North American fixed access networks [Source: Sandvine]

2. RELATED WOK

File or object searching through near endless amounts of movie files, music files, text files, software downloads in a P2P overlay network is a serious challenge. The problem is the unavailability of high quality metadata to improve the search as described in [4]. Many recent works have proposed how socially inspired mechanisms, based on “tags” might be applied in P2P overlay networks to maintain high cooperation between peers even when they act selfishly. Heymann *et* al. [5] discussed methods for generating taxonomy-like relations among tags, based on statistical measures. Andrea *et* al. [6] used replication and mutation in their socially inspire

Fokker *et* al. [7] proposed a P2P Wikipedia based on personalized tag-based navigation system for sharing multimedia content including movies, music, and software over the Wikipedia. Their approach automatically calculated tags from HTML content. In their work, cooperative peers are also incentivized to optimize the overall system performance. Andrew *et* al. [8] proposed building social groups using user interests in P2P networks. Using knowledge discovery techniques, their approach allowed users to connect to each other who are sharing similar type of files although the contents of the files do not overlap. Tempich *et* al. [9] proposed a query routing strategy that considers query traces to create a human social network. It allows a peer to remember another peer who has successfully answered queries for all future query routing decisions. Olaf *et* al. [10] described Tagster, a distributed content sharing application with collaborative annotation (tagging) of contents amongst peers in a P2P network. A peer can tag contents locally as well as can observe the tags globally (assigned by other peers) through a distributed indexing structure although not all peers are permanently online.

3. JXTA FRAMEWORK

Early developers of P2P applications had the difficulty of creating the pipes between peers for communication without centralized authority. This task has been made easier by the developers at Sun. They created JXTA specification using Java which provides protocols for the peers to communicate with each other [11]. Currently there are six protocols in the specification. Peer resolver protocol which enables to query and respond. Peer discovery protocol which helps to advertise and discover content. Peer information protocol that helps to know a peer's status. Peer Binding Protocol that is used to create communication between peers. Peer Endpoint Protocol is used to find the route from one peer to another. Rendezvous Protocol is needed to propagate messages in the network. JXTA framework shown in Fig.2 is used in this work for the P2P topology construction, communication and routing.

JXTA search is a decentralized search engine for P2P overlay networks. Peers can publish a description of the queries they are willing to answer for other peers who are consumers in the overlay network. Both provider and consumer peers are modeled as simply peers in an arbitrary network. In JXTA search peers can act as hubs and can register with other hubs as information providers to field queries from other peers based on arbitrary content description registrations [12].

Fig.3 shows a JXTA search network where each peer interacts with one hub. Each hub forwards requests to registered providers, which might be either another peer or a hub in the network. Each peer in the JXTA network may act as a consumer, provider, and a hub.

4. DESIGN RATIONALE

In this section, we describe our design rationale. The terminologies used in our approach are (1) Peer which shares the content is termed as content's owner peer. (2) Peer which downloads the content is called downloader peer. (3) Peer which queries is called as querying peer.

Every peer is identified by a unique id which is MD5 hash [13] generated using the corresponding group name (of the peer in JXTA) and the peer name. This unique id enables multiple peer groups using the same search service to coexist without overlapping amongst each other. The owner peer tags the content that it wants to share amongst others with the keywords only at the time when content is initially shared. This restriction is placed in our design to reduce the bias of the user. Gradually downloader peers tag the content as they download and use it. This choice is justified as good reviews

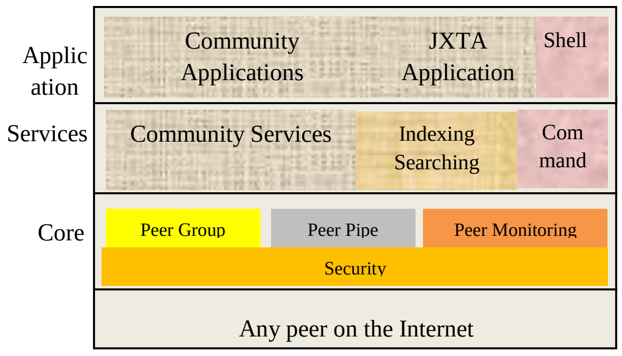


Fig 2: The JXTA Peer-to-Peer Architecture

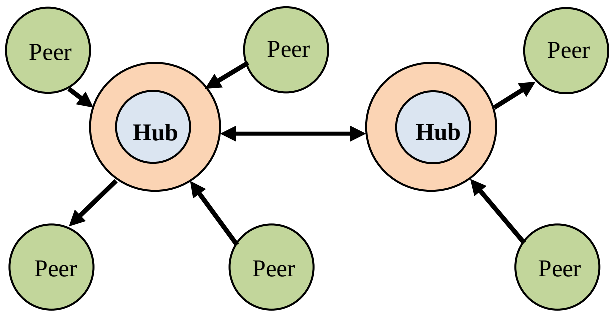


Fig 3: A JXTA Search Network

are obtained from consumers of the products and here, the consumer is the downloader peer. Like owner peer, downloader peer can only tag the content for the same reason. It is assumed that the peer name is unique for a peer group. In our approach, the metadata generated in downloading and tagging the content provides an unbiased definition to the content. This metadata is used in the searching of the content as described in Section 5.1. In tag based search for P2P networks [6, 7, and 10], the relationship between tags and users is very minimal as the file can be tagged by every peer in the network. In our approach, the feature of tagging the file by every downloader peer and the owner, and the provision to rate the previous tags (through associability number) help the search algorithm to create metadata that is not biased as the tagging is voted by the users who use the file.

For ease of implementation, we have assumed that peers share the files stored in a folder. Also, every folder that is shared by owner peer is represented by an XML file that has a structure as shown in Fig.4. In Fig.4, TagPeer node contains the identities of the five recent downloader peers. Here, FID attribute of the file is the MD5 hash of the file path and peer name and hence, FID attribute is unique to every peer in the overlay regardless of the file name.

4.1 Tag storage

Every peer in the overlay maintains a table for storing the tags. An example of such a table is given in Fig.5. In Fig.5, the entry '1' is the unique id of the peer in the corresponding peer group which is MD5 hash of the peer name and group name. The second column i.e., entry '2' is the unique id of the corresponding file which is MD5 hash of file path and peer ID. Entry '3' represents the number of users who have tagged the file. Entry '4' is the corresponding name of the file. Entry '5' is entry point (pointer) to a 'trie' data structure which stores the associated tags and roots of words of file name. The entry '6' points to the recently changed tag in the entry. The entry '7' gives size of the file and '8' gives the format in which the file is saved. Entry '6' makes easy the display of the tag cloud by enabling easy access to the tags as end nodes of tags points to one another. This also helps to distinguish between the words of filename as they are not pointed by the end nodes of the tags. Trie enables search in *O* (m) time, where m is the length of pattern to be search. The preprocessing cost is justified as the numbers of updates are usually low when compared to search operations. Entries '4', '7', '8' are displayed in the results to help the user to choose their interested results. Entry '3' is used for calculating U_{Assoc}. Entries '1' and '2' are used in processing of results of a query.

4.2 Associability number

All the tags to a file are not of equal value. Hence weights are assigned to them by the user. We have assumed font size of the tag as the measure of these weights. Further, the font size is scaled down to integers ranging from 1 to 8. The measure on this scale for a tag is called its associability number. For example a tag with associability number '4' is of more value as a tag than a tag with an associability number '2'. As default the associability of words in file name is taken as 8.

4.3 Trie

In Fig.5, the entry '5' points to a variation of patricia trie [14] which stores the tags that are associated with the file and words in file name. This version of trie is similar to the patricia trie or radix trie except that each branch ends with a node containing a word, its associability number and a pointer to another such end node, if the branch of the trie ending at the node represents a tag. In this work, Trie is used for searching root words obtained from stemming the tags assigned using Lovins stemming algorithm [15]. In information retrieval, stemming is the process of reducing inflated words to their stem or root form e.g., a stemming algorithm might reduce the words "stemmed", "stemming", "stemmer" to the base "stem". In our tag-based approach, stemming is required because the user may use different forms of the same word while querying.

5. IMPLEMENTATION

The algorithms developed in implementing the tag-based search over P2P overlay using JXTA framework are described in this section.

5.1 Querying

In our work, the querying peer queries for a file by providing a set of keywords. A message containing the roots of the keywords, peer id of the querying peer and a number known as associavity threshold (T_{Assoc}) is sent to all the peers by the querying peer. Here, the parameter 'T_{Assoc}' is a tunable parameter specified by the user with a value ranging from 0.125 to 1. This parameter acts as a filter for searching the content i.e., higher the T_{Assoc} value, more refined will be the search results. The algorithm for sending a query is shown in Fig.6. Filtering the results enables the user to choose how much refined the results of the query should be. Results with T_{Assoc1} are subset of results with T_{Assoc2} if $T_{Assoc1} > T_{Assoc2}$. Therefore as higher T_{Assoc} would mean more irrelevant results would appear taking more bandwidth and more time for processing, hence filtering is beneficial. Minimal value of T_{Assoc} is minimal V_{Assoc} which is calculated when all the words of the query are found in tags with minimal associability number which is 1 in this work.

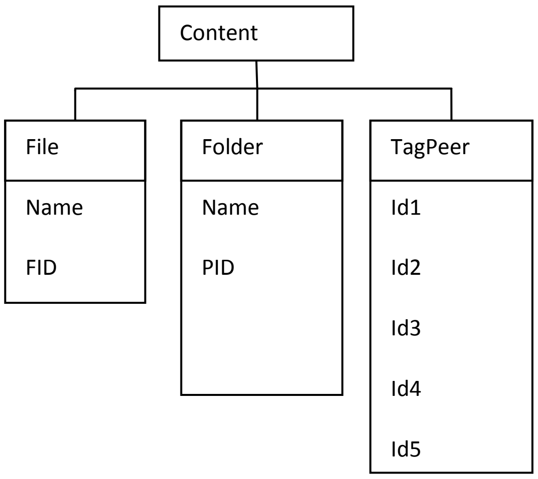


Fig 4: The structure of XML files for a folder

5.2 Searching

The associavity value of the file with respect to the given query is calculated for each table entry using equation 1 as shown below. We describe here an example of searching:

Let A = {$a_1, a_2, \ldots, a_n$} be set of roots which are obtained by stemming n keywords in the query. Let w be number of roots of words of the file name belonging to A. If w is not equal to n, the data structure trie is searched for remaining elements in A. For each root a_i, its associability number b_i is found from the trie. If all the remaining roots are present in the trie, then associavity value (V_{Assoc}) of the file with respective to the query is calculated as:

$$V_{Assoc} = \left(\frac{8w + \sum_{i=0}^{n} b_i}{8n}\right) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

If V_{Assoc} of the file is greater than the associavity threshold i.e., ($V_{Assoc} > T_{Assoc}$), then the corresponding content or file is included in the results.

For each file which meets the query specifications, file name, FID (MD5 hash of the file path and peer id), QPID (peer id of owner peer) is sent to the querying peer as a result of successful search. The search algorithm is given in Fig.7.

Input: Query (q) and associativity threshold (T_{Assoc})

```
//q: array of strings of length n
foreach keyword q_i in 'q' do
    extract root of the word a_i;
end
foreach peer in peer group do
    send querying peer id;
    send array of roots, A;
    send associativity threshold, T_Assoc;
end
```

Fig 6: Pseudo code for querying the group

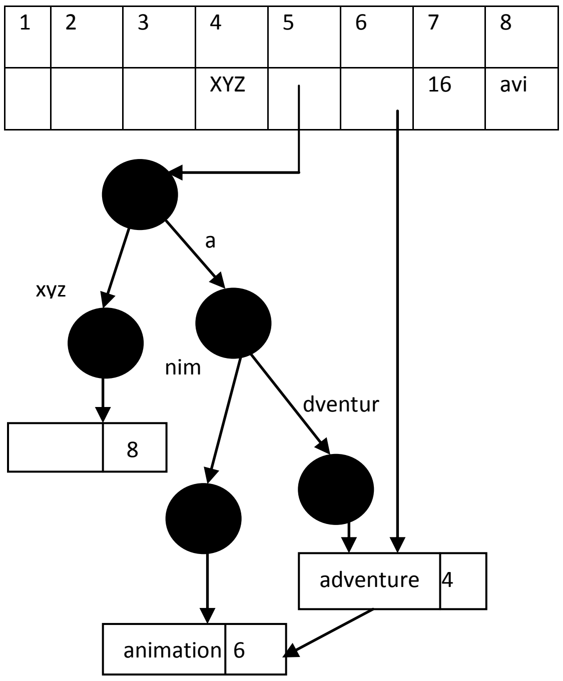


Fig 5: Example of an entry in the table

5.3 Downloading

The messages arriving from the peers in the overlay are processed to remove redundancy and remove entries whose owner peers are unavailable. The redundancy is removed by seeing to that the set of files have unique FIDs. This is achieved here by using hash table with FIDs as key mapping to corresponding PIDs. When a file is selected for downloading, a message containing FID is sent to the owner peer. Owner peer responds to this download request by sending the file to the querying peer. As soon as the download is finished a certificate is sent to the downloader peer from the owner peer marking it (the certificate) as a proof of download. The downloader peer then could tag the associated file within a user defined time span. A similar certificate is sent to the owner peer along with the time span. Digital certificates are present only in the owner peer and downloader peer in the period of time span. Time span is needed as the user might not be interested in tagging the file immediately after download and leaving them without tagging would waste memory resources heavily.

5.4 Storing tags

When a downloader peer tags the file, the tags are sent with the id of the downloader peer to the owner peer along with the digital certificate signed by the downloader peer. The encryption in digital certificate is done using RSA algorithm. The owner peer tries to match the digital signature with the certificates of the downloader peer of a file. If a match is found it accepts the updates otherwise it discards the message. This would ensure the authenticity of the tags.

Input: Querying peer id(QPID),A, and T_{Assoc}

```
if (QPID==PID) then
   //PID is peer id
   discard search;
end
foreach entry in TagStorageTable do
  //B is array of strings
  B ← roots of the words in filename
  W ← 0;
  H ← 0;
  foreach element in array A do
     search B for the element e;
     if (found) then
     w ← w + 1;
     h ← h + 8;
     remove(e,A);
     if (nextEntry==null) then
     exit;
     else
        go to nextEntry;
     end
     end
  end
  if (w==n) then
  add the file to fileSet (S);
  else
  foreach element in array A do
    Search the trie for element,e;
    if (found) then
    h ← h + associability number;
    remove(e,A);
    end
  end
  end
  if (empty(A))then
  V_Assoc ← h/8n(A);
  if (V_Assoc > T_Assoc) then
  add file to fileSet(S);
  end
  send fileSet to querying peer;
end
```

Fig 7: Pseudo code for tag-based search

Consider a file for which the associated XML file is shown in Fig.4. Its' tag peers are id5, id4, id3, id2, and id1. If a peer with id6 downloads the file, id6 will be added to the set of tag peers and the oldest downloader peer in the set which is id1 is removed from the set. A message is then sent to the peer associated with id1 to remove the table entry of the respective file. Another message is sent to update the tags to the remaining four ids along with the updated tags and the associated associability number. If there are new tags in the updated tags, then stemming algorithm is used to extract the root of the tag. Tag Associability is then updated. Let U_{Assoc} be the associability number to be stored in the trie. Let N_{Assoc} be the associability number that is assigned currently. Let D be the number of times the file is downloaded until recently. Let O_{Assoc} be the associability number that is stored in the trie. We safely assume it to be 0 for new tags. Then the U_{Assoc} is then calculated using equation 2 as shown below:

$$U_{Assoc} = \left\lfloor \frac{((OAssoc)*(D-1) + (NAssoc))}{(D+1)} \right\rfloor \ldots\ldots(2)$$

```
Input: List of tags and associability number (L), Download file FID (DFID)
foreach entry in TagStorageTable do
  if (DFID == FID) then
    $N_{Assoc} \leftarrow$ assocNo(L);
    foreach tag do
    root $\leftarrow$ extract(tag);
    found searchTrie(root);
    if (!found) then
    $O_{Assoc} \leftarrow$ 0;
    D $\leftarrow$ 0;
    Calculate $U_{Assoc}$ ;
    addToTrie(root, $U_{Assoc}$);
    else
    $O_{Assoc} \leftarrow$ assocNo(Trie);
    D=findDNO(Trie);
    Calculate $U_{Assoc}$;
    if ($U_{Assoc}$ >= 1) then
      UpdateTrie(root, $U_{Assoc}$);
    end
    end
    end
  end
end
```

Fig 8: Pseudo code for tag storage

The Tag storing or updating procedure is described in Fig.8. If the root of the tag is already a part of the trie and U_{Assoc} is greater than or equal to 1 then its associability is updated in the trie .If the keyword under consideration is not found in the trie, the keyword is inserted into the trie. Every time the peer joins the peer group, the digital certificates in the peer are removed when their life time which is the user defined parameter is expired.

6. SIMULATION EXPERIMENTS

We evaluated the effectiveness of our tag-based search algorithm over JXTA framework by carrying out number of experiments with varying parameters. We have used notations Q, K (Q), R (Q), and A to explain our results. Let a query be denoted by Q. Then the keywords belonging to query Q is set of strings, K (Q). A file, 'f' belongs to the result of query, R (Q) if it satisfies query Q. In tag based search every query is propagated through flooding. In query flooding a query is flooded by the peer to its immediate nodes and the immediate nodes in turn flood the query to their neighbors till hop limit is reached. Use of query flooding for query routing is present in Gnutella [3]. Another search is considered which doesn't use tags for searching and relies only on file names i.e., in this search a file belongs to R(Q) if every k belonging to K(Q) is a substring of file name. Even in this search queries are routed through query flooding. A TTL of 7 which is its maximum value is used in our simulations. We refer to this latter search as flooding in the simulation results, the earlier one as Tag based search that we have developed in this work.

To compare both the searches, the proposed and flooding, 100 files (text, audio, and video) are made available as sharable files in each peer. An array of 500 words is taken and file names are generated by generating random numbers to determine the words in the file name from the array. To simulate tagging, random numbers are again generated to determine the tags and their associability number. A query, Q is generated in similar manner. To ease the data collection, number of tags was limited to 10. Initially 20 peers are taken into consideration. The number of files in R(Q) are collected for 500 queries and their average for a query is taken for comparing the performance. A graph is plotted with this average on Y-axis and total number of tags in 20 peers on X-axis as shown in Fig.9. To estimate the effect of number of peers on number of search results, similar graph (Fig.10) is plotted with 30 peers to see the scalability of our approach. In these simulations, the parameter 'T_{Assoc}' is taken as 1.

From Fig.9 and Fig.10, it can be inferred that for any number of tags, improvised tag-based search proposed in this work provides more search results in terms of number of peers having the file than flooding. It can also be concluded that as the number of tags assigned to files increases, the results provided by the improvised tag-based search for the same set of files also increases. As the number of peers increase in the overlay, the number of search results also increases.

As peers can join and leave the overlay network anytime, the dynamic nature of Peer-to-Peer network has effect on the performance of the search. To determine the impact of churn i.e. peer mobility, we plotted a graph with the average number of files in R (Q) for 500 randomly generated queries produced on Y-axis and percentage of peers out of 20 peers that have completed the processing of query on X-axis. In addition to determining the impact of number of previous downloads on the results, the graphs are plotted for two different number of downloads per peer, 5 and 7. Graphs plotted are shown in Fig. 11. Here also, T_{Assoc} is taken as 1 during the entire process.

From Fig.11 it can be observed that, as the number of downloads by downloader peers increase, the effect of peers leaving the network decreases on the search.

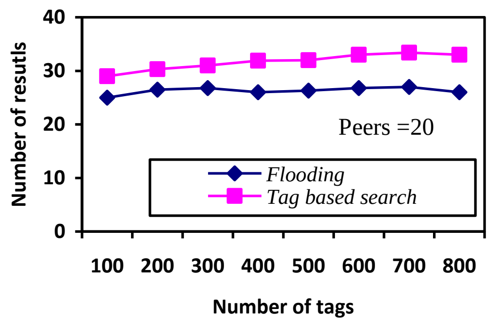


Fig 9: Number of results Vs. Number of tags (20 peers)

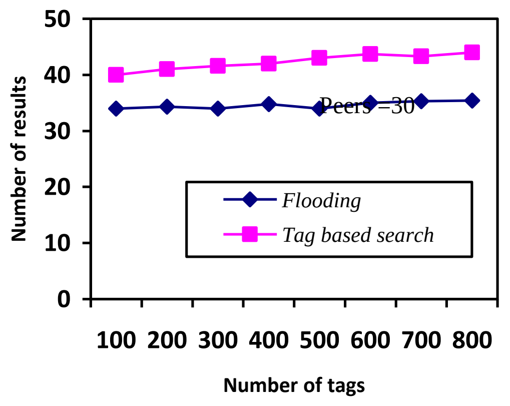


Fig 10: Number of results Vs. Number of tags (30 peers)

To show the effect of ‘T_{Assoc}’ on the number of search results, a graph is plotted with the average of number of files in R (Q), for 500 randomly generated queries produced in 20 peers on Y-axis and T_{Assoc} on X-axis as shown in Fig.12. As shown in Fig.12, it can be inferred that the tag-based search produced more results in comparison to flooding as the value of ‘T_{Assoc}’ decreases. Because of this behavior of tag-based search, we can safely hold the inferences drawn from Fig.9 and Fig.10 as true for all values of the parameter ‘T_{Assoc}’.

7. MATHEMATICAL ANALYSIS

Let T_0 be the associability number of the tag before a download. Let T_0 belong to an integer ranging from 1 to 8. Let X be the associability number assigned to the tag by the downloader peer after it is downloaded. Then, X $=\lfloor Y \rfloor$, Y ε (1,9) where Y can be approximated to a normal distribution [16] with mean value of T. Let after n assignments of the tag, associability number is a random variable Z_n.

Then,

$Z_0 = T_0$

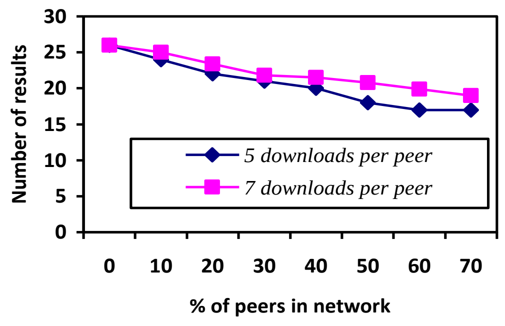


Fig 11: Number of results Vs. % of peers

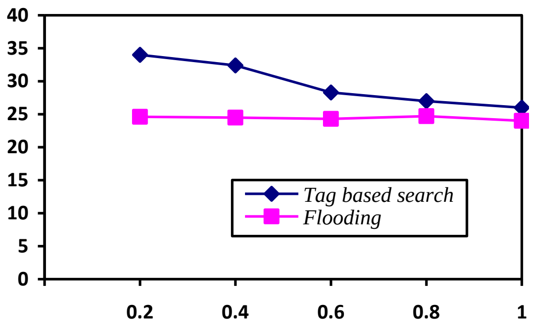


Fig 12: Number of results Vs. Associativity Threshold ‘T_{Assoc}’.

$Z_1 = (Y+Z_0) / 2$

$Z_2 = (2*Y+Z_0) / 3$

In a similar manner, $Z_n = (n*Y + Z_{n-1}) / (n+1)$

Here, Y and Z_n can be approximated as independent random variables. Then Z_n has a normal distribution with mean value as $(n*T+T_0) / (n+1)$. As n approaches infinite (∞), the mean value of the distribution will tend to T. The expected value of the associability of the tag after large assignments will be equal to T. Thus, assigning tags give unbiased definition to the content shared.

The efficiency of the proposed approached is compared with flooding by computing their algorithmic complexity which we explain here in part. For querying, both flooding and tag based search have the same complexities as both use query flooding for query routing as used in this work using JXTA framework. Searching in flooding is nothing but finding whether k keywords are substrings of n strings which when uses suffix tree (as used in this work) for string matching takes *O*(nkm) time to search a file where ‘n’ is the number of files shared, ‘k’ is the number of keywords in the query and ‘m’ is the length of largest keyword. The tag-based search also has the same complexity as it uses flooding for query routing. So, we can conclude that the tag-based search produces more files for the given query in comparison to the standard flooding and has the same complexity, hence improving the search performance in a Peer-to-Peer network.

8. AN EXAMPLE SCENARIO

In this section, we describe an example of tag-based search with screenshots of our simulation run in Fig.13-16. Fig.13 shows an example scenario with owner peer, querying peer and downloader peers. The querying peer becomes the downloader peer when he or she downloads the content from the owner peer.

Fig.14 shows screenshot of the run when a user downloads a file (here for example KungFuPanda2.avi), the tags assigned to the file are shown to the right. The other file names displayed in the screenshot of the Fig.14 are the files available at peer1 and the files which the peer downloaded (also, for which the digital certificate is not yet deleted).

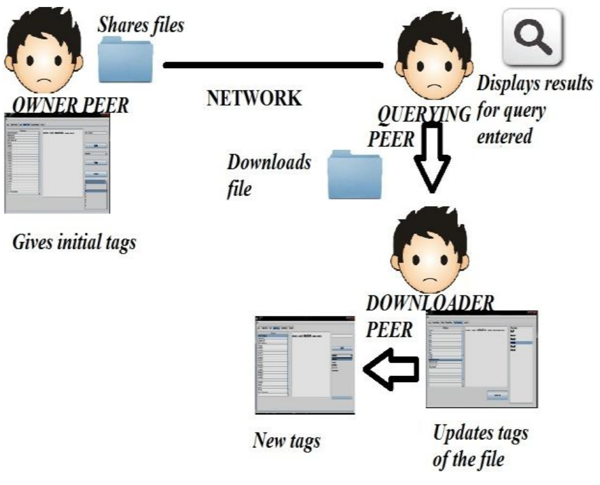
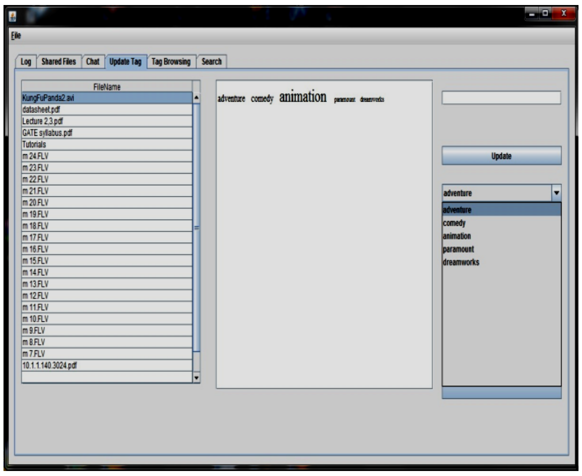


Fig 13: An Example Scenario

When peer1 selects the filename (here, KungFuPanda2.avi), the tags assigned to the file by owner peer and other downloader peers are also displayed in different font sizes which are proportional to their associability number, which is as shown in Fig.14.

After download, a peer can tag a file by typing a keyword in the space provided and its associability number can also be chosen as shown in Fig.15. Here “mark osborne” is typed as a new tag and its associability number is 3 which are decided by the downloader peer. After assigning these two, when the update button is pressed, a message is sent to update the tags as described in our earlier section. To verify whether the tag is updated or not, file list of owner peer is downloaded and the file in consideration is selected from the list as shown in Fig. 16. In Fig.16, peer4 is the owner of the file ‘KungFuPanda2.avi” and “mark osborne” is now displayed with other tags.

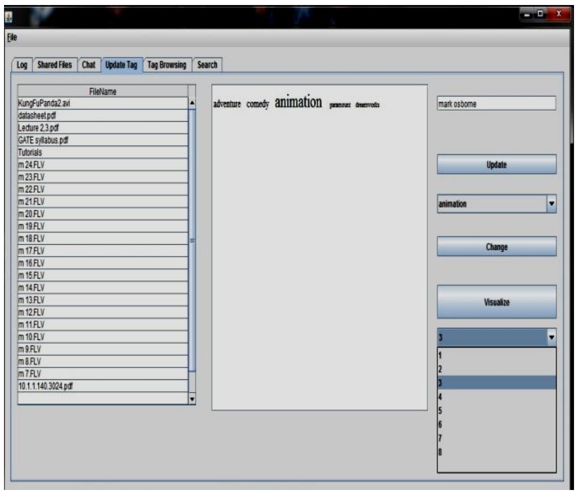


FileName
KungFuPanda2.avi
datasheet.pdf
Lecture 2,3.pdf
GATE syllabus.pdf
Tutorials
m 24.FLV
m 23.FLV
m 22.FLV
m 21.FLV
m 20.FLV
m 19.FLV
m 18.FLV
m 17.FLV
m 16.FLV
m 15.FLV
m 14.FLV
m 13.FLV
m 12.FLV
m 11.FLV
m 10.FLV
m 9.FLV
m 8.FLV
m 7.FLV
10.1.1.140.3024.pdf

adventure comedy animation paramount dreamworks

adventure
adventure
comedy
animation
paramount
dreamworks

Fig 14: Screenshot of Tags assigned to the file ‘KungFuPanda2.avi’ at peer1

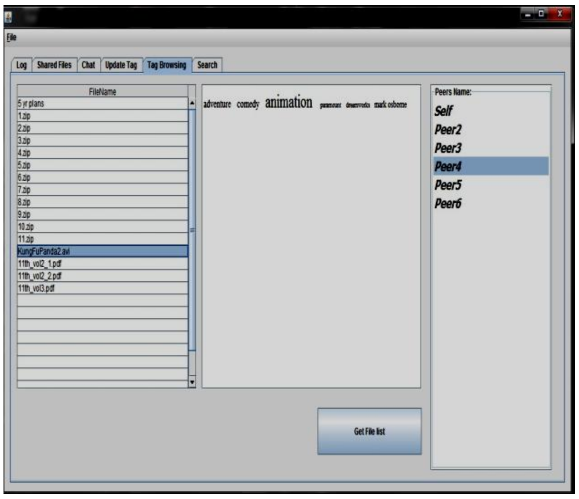


FileName
KungFuPanda2.avi
datasheet.pdf
Lecture 2,3.pdf
GATE syllabus.pdf
Tutorials
m 24.FLV
m 23.FLV
m 22.FLV
m 21.FLV
m 20.FLV
m 19.FLV
m 18.FLV
m 17.FLV
m 16.FLV
m 15.FLV
m 14.FLV
m 13.FLV
m 12.FLV
m 11.FLV
m 10.FLV
m 9.FLV
m 8.FLV
m 7.FLV
10.1.1.140.3024.pdf

adventure comedy animation paramount dreamworks

mark osborne
animation
3
1
2
3
4
5
6
7
8

Fig 15: Screenshot of Tagging the file “KungFuPanda2.avi”



FileName
5 yr plans
1.zip
2.zip
3.zip
4.zip
5.zip
6.zip
7.zip
8.zip
9.zip
10.zip
11.zip
KungFuPanda2.avi
11th_vol2_1.pdf
11th_vol2_2.pdf
11th_vol3.pdf

adventure comedy animation paramount dreamworks mark osborne

Peers Name:
Self
Peer2
Peer3
Peer4
Peer5
Peer6

Fig 16: Screenshot of Updated tags

9. CONCLUSION

In this paper, we have presented a novel tag-based search using peer collaboration, where peers assign tags that correctly describe the type of content they have received from other peers in the network as per their experience with the downloaded content. Our approach is analogous to the way various customers give their opinion about a vendor while transacting with them. The tags assigned by various peers as the content is downloaded by many peers are then used to create an unbiased definition of the content. The JXTA search is modified to consider the tags that are assigned by various peers. Our simulation results show that the proposed tag-based search is better over simple flooding while searching for files in Gnutella like Peer-to-Peer networks. We also mathematically argue that that the ratings by other users help in providing unbiased weights to the tags.

We plan to extend this work by providing a form of social bookmarking in P2P content sharing applications. We also plan to profile individual taggers and thus provide a search unique to an individual.

10. ACKNOWLEDGEMENT

This work was supported in part by Grant number 39/118 (2010) for Scientific research under Major Research Project grant from University Grants Commission, New Delhi, INDIA.

11. REFERENCES

[1] Schollmeier, R. 2001. A definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. In Proceedings of P2P'01, pp. 101.

[2] Napster. Available from: http://www.napster.com

[3] Gnutella protocol 0.6. Available from: http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

[4] Dick C. A. Bulterman. 2004. Is it time for a moratorium on metadata?. IEEE MultiMedia. Vol. 11, No. 4, pp. 10–17.

[5] Heymann P., and Garcia-Molina, H. 2006. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10. Stanford.

[6] Andrea M., David H., Gian P J., Stefano A., and Ozalp B. 2005. Tag-based Cooperation in Peer-to-Peer Networks with Newscast. Technical Report. June 2005, University of Bologna, Italy.

[7] Fokker J., Pouwelse J., and Buntine W. Tag-Based Navigation for Peer-to-Peer Wikipedia. 2006. In Proceedings of Collaborative Web Tagging Workshop. Edinburg, Scotland.

[8] Andrew Fast, Jensen D., and Levine, B N. 2005. Creating social networks to improve peer-to-peer networking. In Proceedings of KDD. pp. 568–573.

[9] Tempich, C., Staab, S., and Remindin, A W. 2004. Semantic query routing in peer-to-peer networks based on social metaphors. In Proceedings of WWW. pp. 640–649.

[10] Olaf Gorlitz, Sizov, S., and Staab, S. 2008. Tagster - tagging-based distributed content sharing. In Proceedings of ESWC'08, pp. 807–811.

[11] Gradecki Joseph D. 2002. Mastering JXTA:Building Java Peer-to-Peer Applications. John Wiley & Sons.

[12] Steve W., David M D., Gene K., and Yaroslav F. 2002. Distributed Search in P2P Networks. IEEE Intenet Computing. pp. 68-72.

[13] Rivest R. The MD5 Message-Digest Algorithm. 1992. RFC 1321. MIT LCS & RSA Data Security Inc.

[14] Morrison, Donald R. 1968. PATRICIA-Practical Algorithm To Retrieve Information Coded in Alphanumeric. Journal of the ACM (JACM).

[15] Lovins, Julie Beth. 1968. Development of a Stemming Algorithm. Mechanical Translation and Computational Linguistics. Vol. 11, Nos. 1 and 2, March and June 1968.

[16] Normal Distribution, Available from: http://en.wikipedia.org/wiki/Normal_distribution