NEIGHBOURING NODE FEATURE COMPUTATION SCHEME TO IMPROVE PEER-TO-PEER NETWORK EFFICIENCY

Naveen Vijay[1], Noor Saba[2], Silpi Dutta[3], Suprith S[4], Suma R[5]

[1]Student, Computer Science and Engineering, T.John Institute of Technology, Karnataka, India
[2]Student, Computer Science and Engineering, T.John Institute of Technology, Karnataka, India
[3]Student, Computer Science and Engineering, T.John Institute of Technology, Karnataka, India
[4]Student, Computer Science and Engineering, T.John Institute of Technology, Karnataka, India
[5]Assistant Professor, Computer Science and Engineering, T.John Institute of Technology, Karnataka, India

Abstract

This project focuses on improving the QoS provided by the searching mechanism in unstructured Peer-to-Peer network. In an unstructured Peer-to-Peer network, any node in the network act as both the server and a client, i.e., there is no centralised server to ping if any node requires a file which is available somewhere in the network. Consequently, a node will send a file query to its neighbouring node(s), and the neighbouring node(s) will forward the query to their neighbour(s), and the process repeats till the query reaches the node containing the required file, which is sent back to the querying node. Our projects aims at improving this system by introducing a probability based statistical selection algorithm, in contrast to the existing blind search methods. In this method of searching, a querying node will send the file query to only one selected neighbour, and a similar forwarding process takes place. The selection process is performed by calculating certain features of each neighbour and selecting the node which more likely has the required file, using standard deviation and probability. This helps improving transmission efficiency and reducing network congestion, with a reasonable increase in the processing overhead.

Keywords: *Peer-To-Peer Network, Neighbouring Node Feature Computation Scheme, Search Mechanism.*

--***--

I. INTRODUCTION

The Peer-to-Peer network is an emerging communication model where a group of systems are interconnected in such a way that all systems have similar service capabilities. This essentially means that every node in the network is a client as well as a server. This kind of decentralized system distribution allows [1] advanced capabilities such as anonymous routing of network traffic, massive parallel computing environments, distributed storage and other functions. Peer-to-Peer networks are infamous for being widely used for pirated software distribution and media sharing, and hence, is often a subject of controversy.

There are two main types of Peer-to-Peer networks; Structure Peer-to-Peer networks and Unstructured Peer-to-Peer networks. A Structured Peer-to-Peer network is one where the network overlay is arranged in a specific structured topology. This allows network communication and transactions to occur in a predictable and efficient manner, since the network topology is predetermined.

An Unstructured Peer-to-Peer network is one where there is no specific structure in the network arrangement. This type of Peer-to-Peer network is more commonly used since it is inexpensive as well as much more easier to set up. However, the unstructured nature of this network makes network operations a lot more unpredictable and complex as compared to a Structured Peer-to-Peer network.

Our project deals with the how searching operations are performed in an Unstructured Peer-to-Peer network. A typical Unstructured Peer-to-Peer network has no centralized server to ping when a peer is required to search for a file, since these files are typically distributed across the various systems of the network. The usual approach to searching a file in such a network is for the peer to flood the network with query packets and let the packets travel throughout the network until it either reaches the peer(s) containing the required file, or it's *Time-To-Live (TTL)* decreases to zero. The drawback of flooding is that in a large-scale network, this approach results in massive traffic overhead. The logical solution to this problem is to selectively send query packets to those neighbours of the peer which have a higher probability of holding the required file, or which are connected to the peers which have higher probability of holding the required file.

II. FEATURES

We propose a system where the neighbor node selection process is performed based on analyzing certain features of the neighbours of the querying peer and sending to the query packets to the best suited neighbour. The features which are to be analysed are:

- *Processing Ability (PA)*
- *Effective Sharing (ES)*
- *Index Power (IP)*
- *Transmission Efficiency (TE)*

Processing Ability is the measure of how many queries and query responses that a peer can handle. A neighbouring peer with higher Processing Ability would typically be a better choice to send a query request.

Effective Sharing assesses the number of files and the quality of files that are being shared by a peer. Due to the unstructured nature of this system, there may be neighbouring peers which download files from other systems on the network, but do not distribute it to other systems. These types of peers are commonly known as leechers. A peer with high Effective Sharing usually has a better chance of responding to a query request than others.

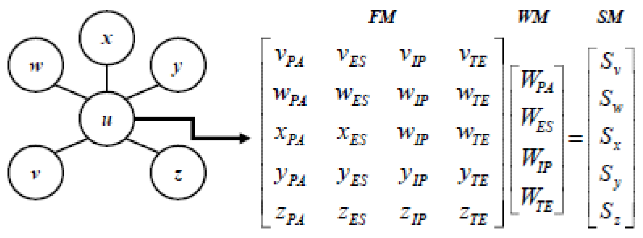
A peer will usually break down a file into smaller chunks before distributing it to other peers. These chunks are labeled by indices, and these indices are stored in the peer's index record. The *Index Power* of a peer assesses the content of these index records are well as the quality of these indices. A peer with a large amount of contents in the index records would typically have a higher probability of matching queries.

The *Transmission Efficiency* measures the quality of transmission between peers by calculating the distance between them. A neighbouring peer which is closer to the querying peer would have a better query response time than peers which are farther away.

These 4 features are calculated for every neighbour of the querying peer and arrange in the form of a n×4 matrix where n is the [2] number of neighbours of that peer. Each row contains the value of one feature for each neighbouring peer. This matrix is called the *Feature Matrix.*

This matrix is then multiplied with another matrix called the *Weight Matrix.* The Weight Matrix is a 4×1 matrix used to determine how much each feature accounts for the final decision of selecting an appropriate neighbour to send a query.

This multiplication computation results in a new matrix called the *Scoring Matrix.* The scoring matrix will be a n×1 matrix, where the row with the lowest value corresponds to the neighbour whose probability of a successful search hit is highest.



FM
WM
SM
w
x
y
u
v
z
v_{PA} v_{ES} v_{IP} v_{TE}
w_{PA} w_{ES} w_{IP} w_{TE}
x_{PA} x_{ES} w_{IP} w_{TE}
y_{PA} y_{ES} y_{IP} y_{TE}
z_{PA} z_{ES} z_{IP} z_{TE}
W_{PA} W_{ES} W_{IP} W_{TE}
=
S_v S_w S_x S_y S_z

Fig.1: Neighbouring Node Feature Computation scheme Matrix Calculation

This system of matrix formation and computation forms the core of our *Neighbouring Node Feature Computation Scheme.* The aim of our project is to successfully calculate the Scoring Matrix for any querying peer, and to prove that this system of selective querying is faster and more efficient than the existing method of flooding.

III. EXISTING SYSTEM

The basic search mechanism used in an unstructured Peer-to-Peer network is flooding. Flooding is a blind search strategy, where the network is flooded [3] with query packets until the query reaches the peer holding the required file, who returns the file to the query sender. Another search approach commonly used is the Random Walk (RW) approach. RW approach sends query messages to random neighbours until one or more neighbours respond. Compared to flooding, it is more efficient as it reduces the network traffic. However, RW approach is fundamentally a blind search technique, since peer selection is random. This method also cannot control the excessive traffic at repeatedly queried peers. This technique is enhanced by the use of Multiple Random Walk (MRW) approach. In this approach, queries are sent to multiple neighbours simultaneously, but these neighbours are selected at random. This approach improves upon RW, but still selects neighbours without any strategies, so there is scope for improvement.

IV. PROPOSED SYSTEM

In this paper, we represent the four features in the form of a matrix. To determine the priority of each feature in the neighbour we use a standard deviation algorithm. This system can be used to select a specific neighbour instead of randomly sending query packets to multiple neighbours, thereby improving the search performance over flooding.

The objective of our paper is to improve the existing search mechanism in an unstructured Peer-to-Peer network by using a neighbouring node feature computation algorithm. The Problem Statements based on this paper are listed below:

- Keeping a track for the number of shared files.
- The quality of the content that is to be forwarded.
- The query transmission between the source and the neighbours.
- The transmission distance between the neighbours.

V. DESIGN

Fig.1 shows the basic architecture of our system. A peer initiates the search mechanism when it wants to get a file on another system in the network. Let this peer be the querying peer. Initiating the search mechanism sends a request to the network server which is monitoring the network. The server then returns the features of the neighbouring peers to the querying peer. The querying peer then constructs the required matrices from these [5] features and selects a neighbour to send a search query. The search query is sent and the neighbour returns the file to the querying peer. Each time a peer shares a file, the features of that peer are updated in the server.

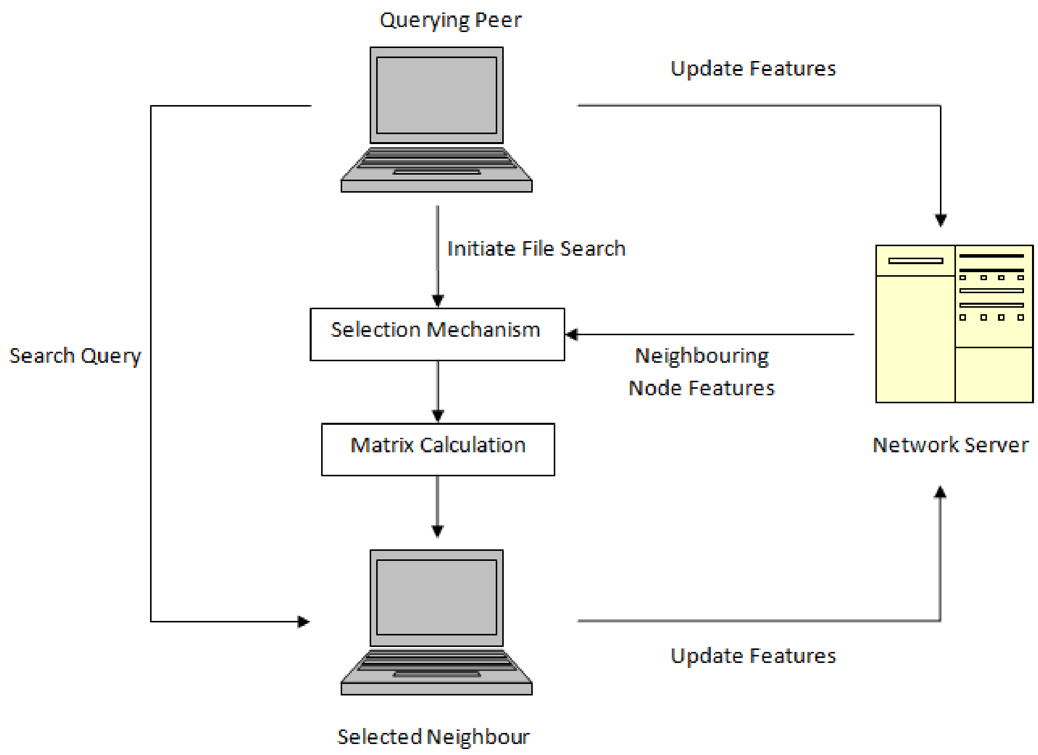


Fig. 2: System Architecture for Neighbouring Node Feature Computation Scheme

Fig.2 shows the flow of control in the system. When a peer initiates the search mechanism, 4 different methods are called to retrieve and calculate each of the 4 required features, namely, “Processing Ability”, “Effective Sharing”, “Index Power” and “Transmission Efficiency”. The 4 features are then sent into a single module to be combined together to form the Feature Matrix. The Feature Matrix is then sent into another module, where standard deviation operations are performed on it to get the Weight Matrix. Finally, the two matrices are multiplied with each other to get the Scoring Matrix. The peer with the highest value in the Scoring Matrix is the selected peer and the search query is sent to this peer.

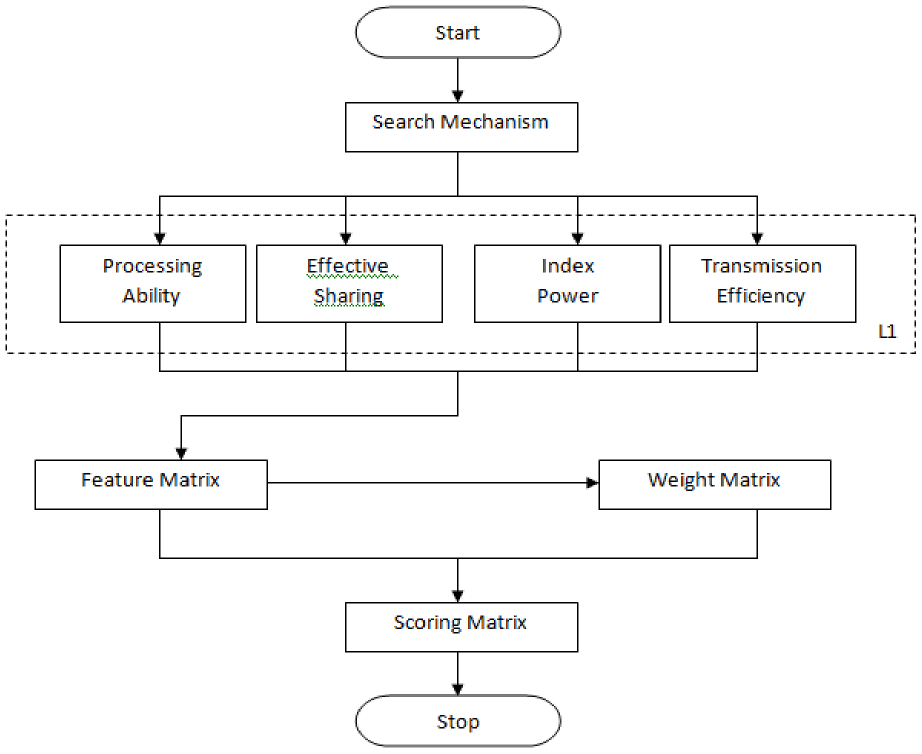


Fig. 3 : Data Flow Level 0 for Neighbouring Node Feature Computation Scheme

VI. Matrix Computation

A. *Processing Ability*

In this module, the Processing Ability of the neighbouring peers is calculated. First, the module sends a request to the network server to retrieve two separate sub-features, namely, the Query Frequency and the Response Frequency. Query frequency is determined by the number of queries that a peer has processed. Response Frequency is determined by the number of queries to which the peer has responded. Once these two sub-features have been retrieved and calculated, they are added to get the Processing Ability. The Processing Ability is calculated for all the neighbours of the querying peer.

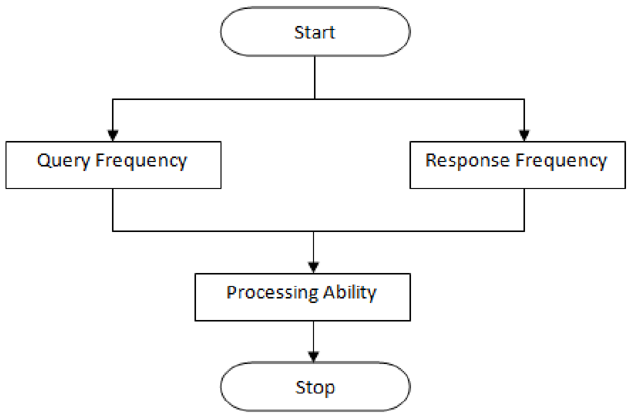


Fig 4: Processing Ability for Neighbouring Node Feature Computation Scheme

a) Query frequency:

$$SQ_1(u) = \sum_{v \in N(u)} NQ(v)$$

$$QMS(u,v) = SQ_1(u) - NQ(v)$$

$$SQMS_1(u) = \sum_{v \in N(u)} QMS(u,v)$$

$$SQMS_2(u) = \sum_{v \in N(u)} SQMS_1(v)$$

$$QF(u,v) = w_1 * \frac{QMS(u,v)}{SQMS_1(u)} + w_2 * \frac{SQMS_1(v)}{SQMS_2(u)}$$

- SQ is the number of queries in total
- QMS is the Query-Score of a neighbour
- SQMS is the sum of query-score of all nodes
- QF is the query frequency of a neighbor

b) Response Frequency:

$$SR_1(u) = \sum_{v \in N(u)} NR(v)$$

$$SR_2(u) = \sum_{v \in N(u)} SR_1(v)$$

$$RF(u,v) = w_1 * \frac{NR(v)}{SR_1(u)} + w_2 * \frac{SR_1(v)}{SR_2(u)}$$

- SR is the sum of response time
- RF is the response frequency

$$PA(u,v) = QF(u,v) + RF(u,v)$$

- PA is the processing ability

B. *Effective Sharing*

In this module, the Effective Sharing of the neighbouring peers is calculated. First, the module sends a request to the network server to retrieve two separate sub-features, namely, the Sharing Count and the Sharing Quality. Sharing Count is determined by the number of files that a peer has shared to other peers in the network. Sharing Quality is determined by the number [6] of files shared by the peer which are queried the most. Once these two sub-features have been retrieved and calculated, they are added to get the Effective Sharing. The Effective Sharing is calculated for all the neighbours of the querying peer.

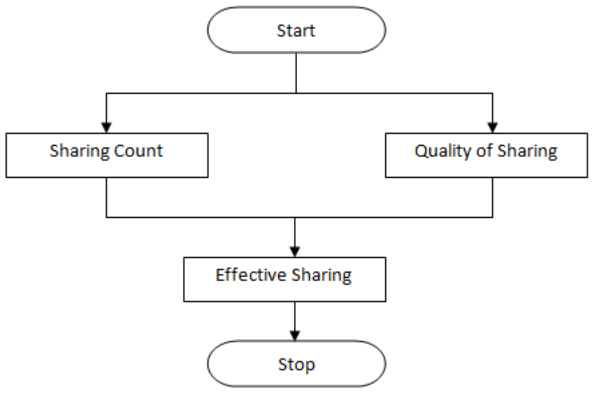


Fig 5: Effective Sharing for Neighbouring Node Feature Computation Scheme

a) **Sharing Count**

$$SF_1(u) = \sum_{v \in N(u)} NF(v)$$

$$SF_2(u) = \sum_{v \in N(u)} SF_1(v)$$

$$SC(u,v) = w_1 * \frac{NF(v)}{SF_1(u)} + w_2 * \frac{SF_1(v)}{SF_2(u)}$$

- SF is the total number of shared files
- SC is the Sharing count of each neighbour.

b) **Quality of Sharing**

$$SFH_1(u) = \sum_{v \in N(u)} NFH(v)$$

$$SFH_2(u) = \sum_{v \in N(u)} SFH_1(v)$$

$$QS(u,v) = w_1 * \frac{NFH(v)}{SFH_1(u)} + w_2 * \frac{SFH_1(v)}{SFH_2(u)}$$

- SFH is the effectiveness of the neighbours of a query peer
- QS is the sharing quality of a neighbour

$$ES(u,v) = SC(u,v) + QS(u,v)$$

- ES is the effective sharing of the neighbour

C. *Index Power*

In this module, the Index Power of the neighbouring peers is calculated. First, the module sends a request to the network server to retrieve two separate sub-features, namely, the Index Count and the Quality of Index. Index Count is determined by the number of [9] messages in a peer's index, i.e., the number of chunks that a file is divided into before sharing. Quality of Index is determined by the characteristics and quality of the files on the peer's index. Once these two sub-features have been retrieved and calculated, they are added to get the Index Power. The Index Power is calculated for all the neighbours of the querying peer.

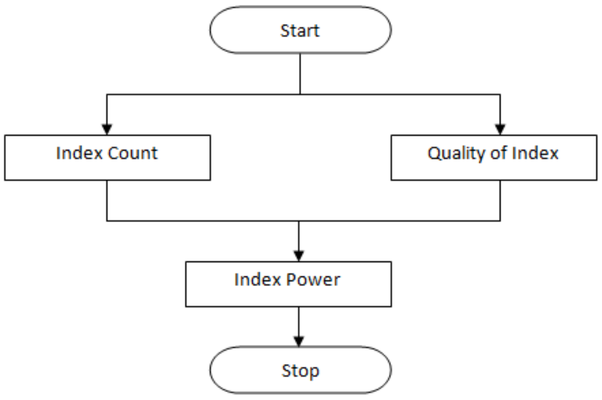


Fig 6: Index Power for Neighbouring Node Feature Computation Scheme

a) **Index count**

$$SI_1(u) = \sum_{v\in N(u)} NI(v)$$

$$SI_2(u) = \sum_{v\in N(u)} SI_1(v)$$

$$IC(u,v) = w_1 * \frac{NI(v)}{SI_1(u)} + w_2 * \frac{SI_1(v)}{SI_2(u)}$$

- SI is the number of indices of the peer that are one hop away
- IC is the index count of the neighbour

b) **Quality of the Index**

$$SIH_1(u) = \sum_{v\in N(u)} NIH(v)$$

$$SIH_2(u) = \sum_{v\in N(u)} SIH_1(v)$$

$$QI(u,v) = w_1 * \frac{NIH(v)}{SIH_1(u)} + w_2 * \frac{SIH_1(v)}{SIH_2(u)}$$

- SIH is the number of indices of the index record which are popular.
- QI is the quality of index

$$IP(u,v) = IC(u,v) + QI(u,v)$$

- IP is the index power of the neighbour.

D. *Transmission Efficiency*

In this module, the Transmission Efficiency of the neighbouring peers is calculated. First, the module sends a request to the network server to retrieve the Link Distance of all the peers. The Link Distance is determined by the length of the transmission [7] link between the querying peer and its neighbours. The Link Distance for each neighbouring peer is normalized to get the Transmission Efficiency. The Transmission Efficiency is calculated for all the neighbours of the querying peer.

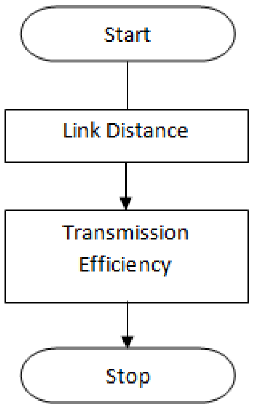


Fig 7: Transmission Power for Neighbouring Node Feature Computation Scheme

$$SLD_1(u) = \sum_{v\in N(u)} LD(u,v)$$

$$LMS(u,v) = SLD_1(u) - LD(u,v)$$

$$SLMS_1(u) = \sum_{v\in N(u)} LMS(u,v)$$

$$SLMS_2(u) = \sum_{v\in N(u)} SLMS_1(v)$$

$$TE(u,v) = w_1 * \frac{LMS(u,v)}{SLMS_1(u)} + w_2 * \frac{SLMS_1(v)}{SLMS_2(u)}$$

- SLD is the sum of all distances of the neighbours
- LMS is the Link-Score of the node
- SLMS is the sum of Link-Scores of the nodes
- TE is the transmission efficiency of the neighbour

E. *Feature Matrix*

The feature matrix is calculated using the algorithm given below:

```
Algorithm CONSTRUCTING_FEATURE_MATRIX(u,$w_1$,$w_2$)
  i.   for each node v $\in$ N(u) do
  ii.      Retrieve the following values NQ, NR, NF, NFH, NI,
           NIH, LD from v
  iii.     Calculate the following terms
```

```
iv.       $SQ_1(u) := \sum_{v \in N(u)} NQ(v)$
v.        $SR_1(u) := \sum_{v \in N(u)} NR(v)$
vi.       $SF_1(u) := \sum_{v \in N(u)} NF(v)$
vii.      $SFH_1(u) := \sum_{v \in N(u)} NFH(v)$
viii.     $SI_1(u) := \sum_{v \in N(u)} NI(v)$
ix.       $SIH_1(u) := \sum_{v \in N(u)} NIH(v)$
x.        $SLD_1(u) := \sum_{v \in N(u)} LD(u,v)$
xi.     for each node $v \in N(u)$ do
xii.      Retrieve the following values: $SQ_1$, $SR_1$, $SF_1$, $SFH_1$, $SI_1$, $SIH_1$, $SLD_1$ from the neighbours of u
xiii.     Calculate the following terms:
xiv.      $SR_2(u) := \sum_{v \in N(u)} SR_1(v)$
xv.       $SF_2(u) := \sum_{v \in N(u)} SF_1(v)$
xvi.      $SFH_2(u) := \sum_{v \in N(u)} SFH_1(v)$
xvii.     $SI_2(u) := \sum_{v \in N(u)} SI_1(v)$
xviii.    $SIH_2(u) := \sum_{v \in N(u)} SIH_1(v)$
xix.    for each node $v \in N(u)$ do
xx.       Calculate the query-score (QMS) and link-score (LMS):
xxi.    $QMS(u,v) := SQ_1(u) – NQ(v)$
xxii.   $LMS(u,v) := SLD_1(u) – LD(u,v)$
xxiii.    Sum of the query-scores of the neighbours of u as follows: $SQMS_1(u) := SQMS_1(u) + \sum_{v \in N(u)} QMS(u,v)$
xxiv.     Sum of the link-scores of the neighbours of u as follows: $SLMS_1(u) := SLMS_1(u) + \sum_{v \in N(u)} LMS(u,v)$
xxv.      Sum of the query-scores of the those peers which are two hops away from u as follows: $SQMS_2(u) := SQMS_1(u) + \sum_{v \in N(u)} SQMS_1(v)$
xxvi.     Sum of the link -scores of the those peers which are two hops away from u as follows: $SLMS_2(u) := SLMS_1(u) + \sum_{v \in N(u)} SLMS_1(v)$
xxvii.  for each peer $v \in N(u)$ do
xxviii.   Calculate the following terms:
xxix.     $QF(u,v) := w_1 * \frac{QMS(u,v)}{SQMS_1(u)} + w_2 * \frac{SQMS_1(v)}{SQMS_2(u)}$
xxx.      $RF(u,v) := w_1 * \frac{NR(v)}{SR_1(u)} + w_2 * \frac{SR_1(v)}{SR_2(u)}$
xxxi.     $SC(u,v) := w_1 * \frac{NF(v)}{SF_1(u)} + w_2 * \frac{SF_1(v)}{SF_2(u)}$
xxxii.    $QS(u,v) := w_1 * \frac{NFH(v)}{SFH_1(u)} + w_2 * \frac{SFH_1(v)}{SFH_2(u)}$
xxxiii.   $IC(u,v) := w_1 * \frac{NI(v)}{SI_1(u)} + w_2 * \frac{SI_1(v)}{SI_2(u)}$
xxxiv.    $QI(u,v) := w_1 * \frac{NIH(v)}{SIH_1(u)} + w_2 * \frac{SIH_1(v)}{SIH_2(u)}$
xxxv.   for each node $v \in N(u)$ do
xxxvi.    Construct the following four columns for the feature matrix FM as follows:
xxxvii.   Processing Ability is the sum of Query Frequency and Response Frequency
xxxviii.  Effective Sharing is the addition of Sharing Count and Quality of Sharing
xxxix.    Index Power is the sum of Index Count and Quality of Index
xl.       Transmission Efficiency is the sum of link distances across 2 hops
xli.    Return FM
```

F. *Weight Matrix*

The weight matrix is calculated using the algorithm given below:

Algorithm *CONSTRUCTING_WEIGHT_MATRIX(FM)*

```
i.      for each column m:=1 to 4 do
ii.       Compute $\overline{C_m} := \frac{1}{n} * \sum_{j=1}^{n} FM(j,m)$
iii.      if n:=1 then
iv.         $S_m := 0$
v.        else
vi.         $S_m := \sqrt{(n-1)^{-1} * \sum_{k=1}^{n} (FM(k,m) - \overline{c_m})^2}$
vii.    for each column m:=1 to 4 do
viii.     $WM(m,1) := \frac{S_m}{\sum_{k=1}^{4} S_k}$
ix.     Return WM
```

G. *Scoring Matrix*

The scoring matrix is calculated using the formula given below:

$$SM(m,1) = \sum_{k=1}^{4} (FM(m,k) * WM(k,1))$$

VII. Non-functional Requirements

H. *Usability*

Simple is the key here. The system must be simple that people like to use it. The user must be familiar with the user interfaces and it must not require a learning curve for new users.

I. *Reliability*

The system should be trustworthy and reliable in providing the functionalities. System failure must be minimized as much as possible.

J. *Performance*

The system should be optimised so that is works efficiently on less powerful systems.

K. *Scalability*

The system should be scalable enough to add new functionalities at a later stage. There should be a common channel, which can accommodate the new functionalities.

VIII. Conclusion

Compared to the flooding search mechanism in dynamic unstructured P2P networks, the Neighbouring Node Feature Computation Scheme is expected to reduce the traffic overhead by more than 80 percent. Moreover, it should achieve a good success rate and shorter response times with reasonable increase in processing throughput. It also helps in reduction of redundant query messages within the network. Further improvement can be made by optimizing the algorithm computation so as to reduce delay and improving response time.

REFERENCES

[1] Design of a Novel Network Architecture for Distributed Event-Based Systems Using Directional Random Walks in an Ubiquitous Sensing Scenario. Cristina Mu˜noz and Pierre Leone, 2015.
[2] Hefeeda, C. H. Hsu, and K. Mokhtarian, “Design and Evaluation of a Proxy Cache for Peer-to-Peer Traffic,” IEEE Transactions on Computers, vol. 60, no. 7, pp. 964–977, 2011.
[3] Hefeeda, and B. Noorizadeh, “On the Benefits of Cooperative Proxy Caching for Peer-to-Peer Traffic,” IEEE Transactions on Parallel and Distributed System, vol. 21, no. 7, pp. 998–1010, 2010.
[4] Making Gnutella-like P2P Systems Scalable,Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, 2009.
[5] Plexus: A Scalable Peer-to-Peer Protocol Enabling Efficient Subset Search, Reaz Ahmed and RaoufBoutaba, Senior Member, IEEE, 2009.
[6] Full-Information Lookups for Peer-to-Peer Overlays, Pedro Fonseca, Rodrigo Rodrigues, Anjali Gupta, and Barbara Liskov, Member, IEEE, 2009.
[7] Where the Sidewalk Ends: Extending the Internet AS Graph Using Trace routes From P2P Users, Kai Chen, David R. Choffnes, Rahul Potharaju, Yan Chen, Fabian E. Bustamante, Dan Pei, Yao Zhao, 2009.
[8] Searching With Multiple Random Walk Queries**,** Santpal S. Dhillon and Piet Van Mieghem**,** 2007.
[9] Free Riding on Gnutella Revisited: The Bell Tolls? ,Daniel Hughes, Geoff Coulson and James Walkerdine, 2005.
[10] Random Walks in Peer-to-Peer Networks**,** Christos Gkantsidis, Milena Mihail, and Amin Saberi**,** 2004.