Journal of Computer Science 7 (3): 434-439, 2011
ISSN 1549-3636
© 2011 Science Publications

Improving Computation Power by Reducing Query Response Time in Peer-to-Peer Environment

[1]Velvizhi Nandagopal and [2] D. Manjula
[1]Department of Computer Applications,
RMD Engineering College, Chennai, India
[2]Department of Computer Science and Engineering,
Anna University, Chennai, India

Abstract: Problem statement: The Peer-to-Peer (P2P) was an emerging model and was being widely adopted in today's internet computing.P2P traffic contributes the largest portion of the internet traffic which was unnecessary and led to delay. In existing Scalable Bipartite Overlay (SBO) network, Out of "N" peers, one group of peers were probing the messages (or) queries whereas the other group of peers were computing. This resulted in high query response time .Since the system was bipartite all peers were not capable of computing and probing, resulting in increased computation time. **Approach:** To overcome the above issue we propose a system RQR (Reduction on query response time) which is decentralized and unstructured where each and every peer can perform both probing and computing with no restrictions. **Results:** Optimal path finding algorithm is designed to find all possible path existing in the P2P network along with the optimal path. **Conclusion:** Our comparison with earlier approaches show that less than 60% of reduction on query response time which increases the system performance.

Key words: Unstructured Peer-to-Peer (P2P), query response time, decentralized architecture, Scalable Bipartite Overlay (SBO), optimal path

INTRODUCTION

A P2P network is composed of an unbounded set of peers. Each peer has its own role. Every peer to peer network uses one of the following architectural formats. Centralized Architecture, Decentralized Architecture and Hybrid Architecture. Among these the decentralized architecture does not contain a central server, which purist administrators would consider a "True" peer to peer network. Decentralized architecture further can be divided into structured, unstructured and hybrid p2p networks .In structured p2p overlays a distributed hash table data structure is used in which every data item can be located within a small number of hops at the expense of keeping some state information locally at the nodes (Demetrios *et al*., 2007). Unstructured p2p systems (Cai and Wang, 2004; Chawathe *et al*., 2003; Punithavathi and Duraiswamy, 2010) are highly infrastructured, because of their decentralized nature we can easily perform updates, increased storage and it offers fault tolerant properties.
The centralized architecture maintains a central global file index for searching, and it is commonly believed that the central index is a single point of failure for the system (Liu *et al*., 2005; Khan *et al*., 2005).

Unstructured p2p networks do not include a strict organization of peers or their content (Gudu and Yuksel, 2009). In existing p2p systems traffic contributes the largest portion of the internet traffic which is unnecessary. To address the limitation of existing works and meet the requirements we built a decentralized unstructured p2p system supporting the following constraints.

- Scalable: The system may be extended based on the request from new incoming peers
- Churn: Peers can enter and leave the network at any time
- Failures: peers can crash at any time without warning other peers (Weiss *et al*., 2010)

Related work: So many techniques have been implemented to reduce traffic in p2p environment. Common Junction Methodology (CJM) that resolves the topology mismatch problem and also reduce the large amount of redundant traffic over the network (Bhushan *et al*., 2010). Based on their measurements of popular p2p systems such as Fast Track (including KazaA and Grokster) Gnutella and Direct connect, the

Corresponding Author: N .Velvizhi, Department of Computer Applications RMD Engineering College, Chennai, India

studies in (Qiu and Srikant, 2004; Ritter, 2001; Li and Chao, 2010; Chen and Liu, 2009) have shown that p2p traffic contributes the largest portion of the internet traffic. Around 18 percent of all Gnutella queries return no results, despite the fact that for atleast two thirds of these queries, the desired results are available in the system. In addition,such queries often suffer long response time (Liu *et al*., 2005; Modarresi *et al*., 2009).

Broadcast-based systems, eg., Gnutella use message flooding to propagate queries. In this the source node sends message to its neighbor, inturn the received neighbor sends message to their neighbors. So there is no specific destination. Every neighbor peer is contacted and forwards the message to its own neighbors. Such systems have been successfully deployed in worldwide adhoc networks due to their simplicity and versatility (Zhu *et al*., 2008; Tang *et al*., 2008).

In Adaptive Connection Establishment (ACE) there is an overlay multicast tree among each source node and the source peer has certain diameter and further optimizes the neighbor connections that are not on the tree while retaining the search scope (Liu *et al*., 2005), which is not scalable.

In clustered p2p network,the nodes are classified into supernode and ordinary nodes.The supernode connect among themselves to form an overlay network just like the nodes in a flat p2p network.A supernode and all ordinary nodes connected to it form a cluster.Each supernode maintains an index of all the objects available in its local cluster (Weiss *et al*., 2010). A superpeer table is maintained by a bootstrap server.Any peer joining the P2P network and wishing to become a superpeer must first issue a request to the bootstrap server (Zhu *et al*., 2008).

In SBO (Scalable Bipartite P2P Overlay Network) design, the nodes are categorized into two in which one category of nodes only probe the messages (or) queries where as the other category of peers only computes (Liu *et al*., 2007). In this technique the group of peers which is assigned for computation purpose will be loaded heavily when compared to the probing peers.Also,Here at initially Bootstraping node assigns the color of the system say red color for probing and white color for computing.

ICRQR doesnot have any Bootstraping server to assign the color and task and there is no such peers like only for either probing (or) computing, Thus every peer in ICRQR has its own roll (Either Bootsraping or client based on the situation). All peers are capable of both probing and computing which increases the system performance and reduces the query response time.

In SBO, initially when a new peer wants to join in the existing network ,each joining peer is randomly assigned either probing peer (or) computing peer. In this case either probing peer may be more than the computing peers in numbers or vice versa. Because of above assumption the throughput of the system goes down. In SBO, Probing peer may have an immediate neighbor with less traffic and query response time but it may not be a computing peer. Here we need to go for other peer which can perform computation, this consumes more time. Thus the response time will be increased. This situation reduces the system performance. In ICRQR if a new peer is joining in the network, then the joining peer does both probing and computing based on the request from other peers and hence the waiting time and processing time will be comparatively less than the existing SBO. This results in Reduction in query response time and increased throughput.

Design of ICRQR: This system is an efficient method to select query forwarding paths and logical neighbors. In ICRQR design, all peers which are connected in the network are in ready state to probe or compute the queries.

The optimal path algorithm finds all the possible paths existing from source to destination along with the optimum path through which the query can be forwarded. After computing all the above, the query is forwared through the optimal path and the time taken to send and receive the queries are recorded as Adjacency matrix. At initially time quantum is assigned along with system time for peers. If any peer is not arrived response within the time quantum given to them, then the queries are resend and the waiting time is calculated along with the processing time.

In Fig. 1, there are 4 peers namely P1,P2,P3 and P4 which are connected through common network. In the above example Peer “P1” is in need of some data which is there in either “P2”,”P3” or “P4”. Since peer P1 is in need of data , “P1”can be considered as source peer and “P1” will have the following details in the monitor display:

- The list of other peers which are connected to source peer “P1”(i.e., IP Address of other peers)
- Time of joining in the network
- Time of leaving from the network

The peers details: In our example, the following details are available in the monitor display of peer P1. IP Address of peer ”P2” (Along with joining time in the network and leaving time),IP Address of peer “P3”(Along with joining time in the network and leaving time) and IP Address of peer “P4” (Along with joining time in the network and leaving time).The new

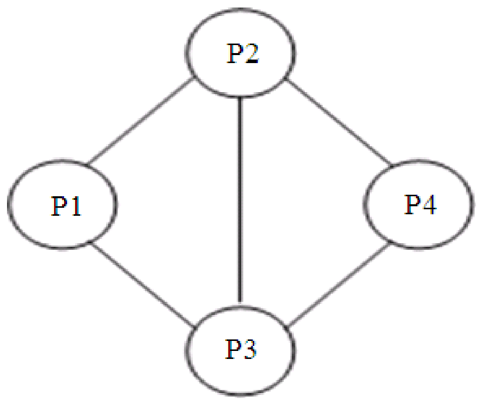

Fig. 1: Example of P2P Network (a)

peer “P5” wants to join in the network,”P5” Should send ping message to other peers which are connected in the existing network. Similarly we can have “N” number of peers. The set of peers which are active will be displayed in the source peer monitor along with the time in which a particular peer joined. Once a peer has joined in the network, it will periodically ping the network connections and obtain the IP address of other peers in the network which can be used to create new connections for the peer’s rejoining. The peer which is crashed cannot be displayed in the monitor so that the updated network alone are available. Thus peers can join and leave the network at any time.

Finding paths: At initially TTL=0, when the message gets hit the time is recorded in Matrix format. If any peer receives the message with same messageID which was already received by the peer then the message will be discarded or updated by the receiving peer. Since duplicate messages severely affect the response time and scalability of P2P systems.

Figure 2a-d shows the all possible paths from source node P1 to destination P4. The set of all possible paths are (1)P1→P2→P4 (2) P1→P2→P3→P4 (3) P1→P3→P2→P4 4) P1→P3→P4. Among these, the path which takes minimum time to hit the query can be considered as the optimal path. To find the optimal path ,the optimal path algorithm is applied. Here in this example the optimal path is P1→P2→P4. Once the optimal path is identified, then the query will be sent in the regular interval and query response time will be evaluated.

Comparison of algorithms: There are algorithms to find the optimal path based on the weights in a given graph. Dijkstra’s Algorithm, is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree. Algorithm starts at the source vertex, S, it grows a tree, T, that ultimately spans all vertices reachable from S. Vertices are added to T in order of distance i.e., first S, then the vertex closest to S, then

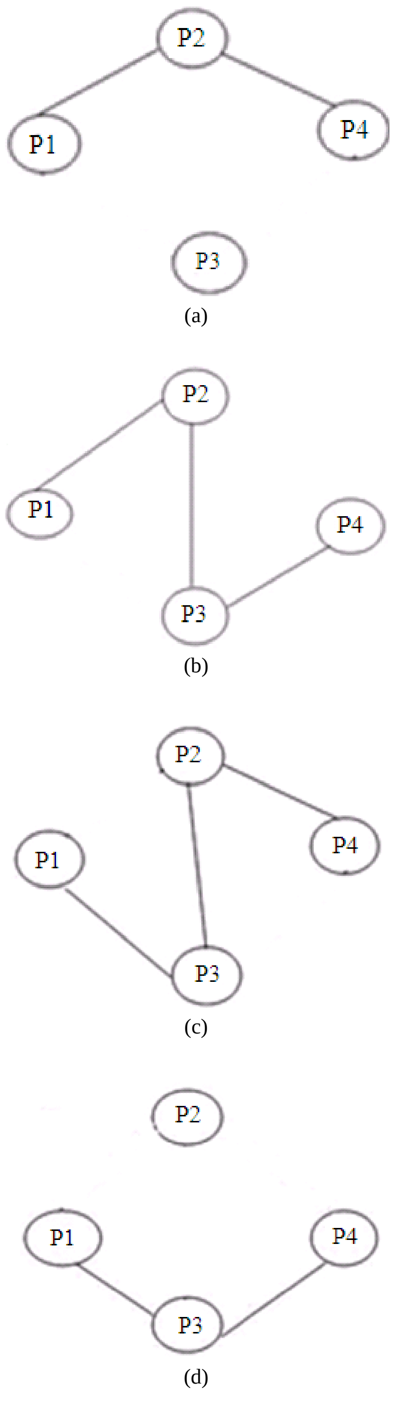

Fig. 2: Set of all possible path

the next closest and so on. This algorithm has complexity of an order of n^2. So it is efficient enough to use for relatively large problems. The major disadvantage of the algorithm is the fact that it does a blind search there by consuming a lot of time waste of necessary resources. This increases the computation time.

PRIM'S algorithm finds a minimum-cost spanning tree of an edge-weighted, connected, undirected graph G(V,E). This algorithm constructs the minimum-cost spanning tree of a graph by selecting edges from the graph one-by-one and adding those edges to the spanning tree. Time taken to check for smallest weight arc makes it slow for large numbers of nodes. This results in high processing time.

Difficult to program, though it can be programmed in matrix form.

Floyd-Warshall's Algorithm is a graph analysis algorithm for finding shortest paths in a weighted graph (with positive or negative edge weights). A single execution of the algorithm will find the lengths (summed weights) of the shortest paths between n all pairs of vertices though it does not return details of the paths themselves. The algorithm is an example of dynamic programming. The complexity of this algorithm is O (n^3).The disadvantage of this algorithm is the inclusion of waiting time along with processing time which degrades the system.

MATERIALS AND METHOD

Optimal Path Algorithm: This algorithm finds the all possible paths from source to destination peer and hence finds the optimal path. Unlike the above mentioned algorithms, this algorithm computes the waiting time and query processing time.The waiting time of a query is comparatively less because the optimal path is already determined and through which we can send and receive the queries. This Algorithm has the complexity of order of O (n^2).

```
1. Initialize TTL = 0
2. Get the Adjacency Matrix [A] for the Network and copy it to path [ ][ ]
3. Get all possible path based on matrix A
4. Procedure Optimal Path ( )
   For k: = 1 to no of vertices
      For I := 1 to no of vertices
         For j: = 1 to no of vertices
            Path[i][j] = minimum ( path[i][j], path[i][k]+path[k][j] );
```

where I,j are the source and the destination ,k is the intermediate node and path[i][j] stores the shortest path.

```
5. Get the system time at the time of sending the query
6. Calculate the Actual Time.
7. Processing time = A[source,destination]+A[destination,source] (Time to send the request and receiving acknowledgment)
8. Waiting time WT (n) = Waitingtime(n-1)+processing time](n-1)(in general):
9. Response time(n) = Processing time(n)+ waiting time(n)
10. Calculate Average Waiting Time(AWT) = $\frac{\sum_{i=1}^{n} WTi}{n}$
11. Average Response time(RT) for "n" number of queries = $\frac{\sum_{i=1}^{n} RTi}{n}$
```

where I = 1,2,3…n.

```
12. In case of not receiving the response apply
       If (Waiting Time == Systemtime)
          If(Response Received)
             If(flag==1)then
                Success(process completed)
             Exit
                End if
          Else Resend the queries
                End if
       End if
```

RESULTS AND DISCUSSION

The monitor display will have the following details:

IP Address of P1(10.0.4.32)8.00AM
IP Address of P2(10.0.4.33)8.02AM
IP Address of P3(10.0.4.32)8.07AM
IP Address of P4(10.0.4.32)8.10AM

Let predetermined periodic Time Interval be TI. At initially 50 Queries were sent in the periodic Time Interval 50, 100 and 150 ms. The response time is recorded for first series. The same process is continued with 100 and 150 queries and the corresponding response time is recorded in the Table 1.

Table 1: Response time reduction with different time interval

S. No:	Number of queries (in 10 sec)	TI = 50ms	TI = 100ms	TI = 150ms
1	5	3.10	2.30	3.42
2	10	4.23	4.43	3.58
3	15	6.20	3.29	3.39
4	20	9.00	3.98	4.11
5	25	12.10	6.18	12.27

Table 2: Comparison of average response time (SBO AND ICRQR)

S.No	Number of Queries (10 sec)	Average Response Time(in millisec) (SBO)	Average Response Time(in millisec) (ICRQR)
1	5	4.01	2.10
2	10	7.23	3.90
3	15	10.45	5.92
4	20	14.83	7.99
5	25	18.75	10.78

Figure 3 shows the result of some samples of TI at 50-150 milliseconds respectively, where x-axis indicates the number of queries and y-axis represents average response time per query in milliseconds. Though the queries are sent at different time interval the response time is not affected during heavy traffic. Similarly number of queries sent per unit time also can be increased based on the requirements and availability. Finally the throughput of the system remains same. The SBO architecture uses Bootstraping peer which acts as a server and it does all process. So, all other peers depend on bootstrapping node and hence the system is not directly communicating each other. This takes much time to complete a process or set of processes. In ICRQR a peer communicates directly with other peer and the result shows that the query is processed through optimal path. Thus the response time of our system is 50% less than the earlier system SBO.

Table 2 show that the average response time (in milliseconds)of SBO and ICRQR.Here the same set of queries are sent for processing with two different P2P design model SBO and ICRQR. The data is arrived in the regular interval by sending queries range from 50 to 250 are recorded in the following format which is showed in Table 2.

ICRQR with SBO: We have discussed the design of SBO to further improve ICRQR. The system is designed with SBO architecture and sent set of queries and arrived the response time. Similarly the same set of queries are sent and recorded the query response time with ICRQR. The data which we arrived through the above mentioned design are presented below.

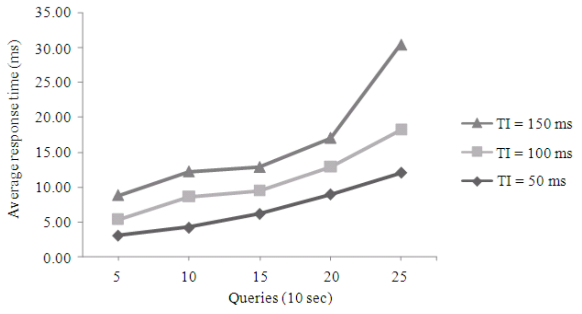

Fig. 3: Response time reduction in dynamic p2p environment

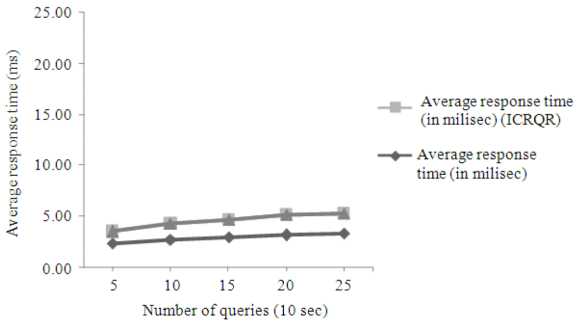

Fig. 4: Comparison on response time

Figure 4 shows the comparitive results of SBO and ICRQR, where x-axis indicates the number of queries and y-axis represents average response time per query in milliseconds. The graph shows that the average query response time of ICRQR is about 50% less compared to SBO. In the Fig. 4, we can see that the convergent speed of SBO is the slowest, so its overall performance in dynamic environments is not as good as ICRQR. Overall, ICRQR outperforms SBO.

CONCLUSION

We propose ICRQR to reduce the query response time in P2P environment. This system is scalable and completely distributed. Also, does not require any global knowledge when a node is optimizing its logical neighbors. The performance benefit of ICRQR is consistent with different time intervals and different amount of queries. ICRQR achieves about 50% of reduction in query response time. Our experimental results show that ICRQR comparatively outperforms existing approach. Further this system can be enhanced with the security, where only the peers which are probing or computing can have the original messages, so that the message is secured from other external causes.

REFERENCES

Bhushan, S., M. Dave and R.B. Patel, 2010. CJM: A Technique to reduce network traffic in p2p systems. Proceedings of the International Conference on Advances in Computer Engineering, June 20-21, Bangalore, Karnataka, India, pp: 306-308. DOI: 10.1109/ACE.2010.55

Cai, H. and J. Wang, 2004. Foreseer: A novel, locality-aware peer-to-peer system architecture for keyword searches. Middleware, 3231: 38-58. DOI: 10.1007/978-3-540-30229-2_3

Chen, H. and Y. Liu, 2009. Difficulty-aware hybrid search in peer-to-peer networks. IEEE Trans. Parallel. Distributed Syst., 2: 71-82.

Demetrios, Z.Y., V. Kalogeraki and D. Gunopulos, 2007. PFusion: A P2P architecture for internet-scale content-based search and retrieval. IEEE Trans. Parallel. Distributed. Syst., 18: 804-817.

Gudu, H. and M. Yuksel, 2009. Limited scale-free overlay topologies for unstructured peer-to-peer networks. IEEE Trans. Parallel. Distributed Syst., 20: 667-679. DOI: 10.1109/TPDS.2008.150

Khan, Z.A., S. Shahid, H.F. Ahmad, A. Ali and H. Suguri, 2005. Decentralized architecture for fault tolerant multi agent system. Proceedings of. Autonomous Decentralized Systems, Apr. 4-8, Rawalpindi, Pakistan, pp: 167-174. DOI: 10.1109/ISADS.2005.1452043

Li, J.S. and C.H. Chao, 2010. An efficient superpeer overlay construction and broadcasting scheme based on perfect difference graph. IEEE Trans. Parallel. Distributed Syst., 21: 594-606. DOI: 10.1109/TPDS.2009.94

Liu, Y. S. Member, L. Xiao and L.M. Ni, 2007. Building a scalable bipartite p2p overlay network. IEEE Trans. Parallel. Distributed Syst., 18: 1296-1306. DOI: 10.1109/TPDS.2007.1059

Liu, Y., S. Member, L. Xiao and L.M. Ni, 2005. Improving unstructured peer-to-peer systems by adaptive connection establishment. IEEE Trans. Comput., 54: 1091-1103. DOI: 10.1109/TC.2005.146

Modarresi, A., A. Mamat, H. Ibrahim and N. Mustapha, 2009. Modeling and simulating semantic social overlay peer-to-peer systems. J. Applied Sci., 9: 3547-3554. DOI: 10.3923/jas.2009.3547.3554

Punithavathi, R. and K. Duraiswamy, 2010. A fault tolerant mobile agent information retrieval system. J. Comput. Sci., 6: 553-556. DOI: 10.3844/jcssp.2010.553.556

Qiu, D. and R. Srikant, 2004. Modeling and performance analysis of bittorrent-like peer-to-peer networks. Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, (SIGCOMM’04), ACM New York, NY, USA., pp: 367-378. DOI: 10.1145/1015467.1015508

Tang, X., J. Xu and W.C. Lee, 2008. Analysis of TTL-based consistency in unstructured peer-to-peer networks. IEEE Trans. Parallel. Distributed. Syst., 19: 1683-1694. DOI: 10.1109/TPDS.2008.44

Weiss, S., P. Urso and P. Molli, 2010. Logoot-undo: Distributed collaborative editing system on p2p networks. IEEE Trans. Parallel. Distributed Syst., 21: 1162-1174. DOI: 10.1109/TPDS.2009.173

Zhu, Z.Z., P. Kalnis and S. Bakiras, 2008. DCMP: A distributed cycle minimization protocol for peer-to-peer networks. IEEE Trans. Parallel. Distributed Syst., 19: 363-377. DOI: 10.1109/TPDS.2007.70732