Research of Replication Mechanism in P2P Network

Dongming Huang, Zong Hu
College of Education
Ningbo University
No.818 Fenghua Road, Ningbo, Zhejiang 315211, China
CHINA
huangdongming@nbu.edu.cn, nbshiny@gmail.com

Abstract: - P2P network is a dynamic self-organization network, in which peer can freely join or leave, so there will lost a lot of important data when some important nodes fail, and there exists load imbalance of node in the P2P network. These features are not good to the expansion of the P2P application, so this paper introduce replicate mechanism, not only enhances the reliability of the network, but also lets the network load balance, at the same time this paper introduce synchronize mechanism to solve the problem of data update consistency in the p2p network.

Key-Words: - P2P Network, Replication Mechanism, Load Transfer, Synchronize Mechanism, Data Update Consistency, Load Balance, Reliability

1 Introduction

Compared to the traditional Client/Server network, P2P[1-3] networks has many advantages, such as scalability, robustness, fault tolerance, so P2P networks attract more and more people's attention, and more and more companies development application software in the P2P platform, for example, file download software---BT, Emule, video streaming media software---Pplive. But in these P2P networks nodes can freely join or leave, and there are more and more network attack coming out, so directly lead to the failure of the important nodes in the network, which led to loss these important data in the failure nodes or not access these important data, finally it has some effect to the entire network users. Although some P2P platforms provide some security mechanisms, but these security mechanisms do not prevent the attacks of network, such as DoS attacks, so in order to reduce the possibility of important data loss caused by the failure network nodes, replication mechanism need to introduce, backup some important raw data in other nodes of the network, this mechanism not only improve the reliability of the network, but also reduce the delay time of user getting resources, making full use of the network resources, and make the network load to balance.

This paper firstly introduce the P2P network model, analysis the problems existing in the P2P network model, and provide the corresponding replication mechanism to ensure data consistency.

The rest of the paper is organized as follows. Section 2 describes the background of the replication mechanism. In section 3 introduce the model of P2P networks. In section 4, we give replication mechanism, detailed describe the process of network node load transfer. Section 5 design the algorithm of implement update data consistency. Finally, the conclusion is given in section 6.

2 Background

2.1 Replication mechanism

Replication related works that have recently been published are [4-6] where the goals are somewhat different; maximizing hit probability of access requests for the contents in P2P community, minimizing content searching (look-up) time, minimizing the number of hops visited to find the requested content, minimizing replication cost, distributing peer (server) load, etc.

Kangasharju et al. [5] studied the problem of optimally replication objects in P2P communities. The goal of their work is to replicate content in order to maximize hit probability. They especially tackled the replication replacement problem where they proposed LRU (least recently used) and MFU (most frequently used) based local placement schemes to dynamically replicate new contents in a P2P community. Maximizing hit probability does not satisfy the required QoA and, furthermore the two different goals lead to different results.

Lv et al. [4] and Cohen and Shenker[6] have recently addressed replication strategies in unstructured P2P networks. The goal of their work is to replicate in order to reduce random search times.

Yu and Vahdat [7] have recently addressed the costs and limits of replication for availability. The goal of their work is to solve the minimal replication

cost problem for a given target availability requirements, thus they tried to find optimal availability for given constraint on replication cost where the replication cost was defined to be the sum of the cost of replica creation, replica tear down and replica usage. Our work differs in that our goal is to replicate content in order to satisfy different levels of QoA values required by individual users. Furthermore, their work does not take P2P system specific features such as changing peers state–going up or down - into account.

Related to supporting lookup services, there are many ongoing research efforts such as Chord [8] and Pastry [9]. They detail the mechanisms for supporting the services that they offer such as indexing, lookup, insert, search, update, and delete. While some of them support fault tolerance by replicating the mapping information, i.e., the key/value binding information on multiple peers, they do not give any availability guarantee for values, e.g., files or multimedia contents, than that of ‘best-effort’ availability support. Furthermore, it is not clear under which criterion the number and location of replicas are determined.

2.2 Data consistency

In P2P systems, there are several conventional works that address the consistency management among replicas. In [10], the authors discuss consistency management in P2P networks by broadcasting invalidation reports using a hybrid push/pull approach. In [11], the authors proposed an update propagation method to replicas based on the gossiping messaging. These approaches are more optimistic in nature, whereas in our approach, the strict consistency conditions at various levels are maintained by broadcasting lock requests on quorums.

In mobile database environments (but not MANET), several consistency management strategies that consider peer disappearance have been proposed [12-15]. Most of these strategies assume an environment where mobile hosts access databases at sites in a fixed network, and replicates/caches data on the mobile hosts because wireless communication is more expensive than wired communication. They address the issue of keeping consistency between original data and its replicas or cached data with low communication costs. These strategies assume only one-hop wireless communication, and thus, they are different from our approach which assumes multi-hop communication in MANETs. Since network partitions frequently occur, different consistency management strategies are required in P2P MANETs. In [15] the authors proposed a formal theory for maintaining temporal and semantic based conditions in terms of broadcast transactions. The idea of maintaining temporal consistency in a group is similar to our TC (Time-based Consistency) described later. However, the focus in [15] was more on time-based transaction consistency. Their idea can be integrated with our TC in the sense that we can use transaction’s temporal consistency condition within a group of peers.

Recently, data replication is becoming more popular and significant topic of research in MANET [16-18]. Several methods have proposed for preserving consistency among replicas in MANETs [19-21]. In [19], the authors proposed methods by which replicas are allocated to a fixed number of mobile hosts that act as servers and keep the consistency among the replicas. In there, the consistency is maintained by employing a strategy based on the quorum system that has been proposed for distributed databases [22]. In [20], the authors extended the methods proposed in [19] by applying probabilistic quorum system [23] and gossip based on message routing [24]. Their methods are considered similar to ours because consistency among replicas is maintained based on the quorum system. However, in [19] and [20], the authors did not assume the strict consistency but aimed to keep the consistency in the entire network. Therefore, the locality in P2P MANETs described in this paper, was not taken into account.

In [21], the authors defined two different consistency levels, local observation consistency and global observation consistency. Global observation consistency is equivalent to GC (Global Consistency) considered in this paper. Local observation consistency is almost equivalent to PC (Peer based Consistency), except that it requires replicas to eventually converge to the most recent version. In [21], only two different consistency levels are defined, whereas in here we define seven levels. Moreover, the authors tried to keep consistency based on an optimistic manner, i.e., transactions are tentatively committed and the consistency is checked afterward by using serializability graphs. Such an optimistic approach may not work well in MANETs because it will cause a large number of aborts and rollbacks of transactions due to conflicts of data operations performed in partitioned networks.

3 Model of P2P Networks

P2P networks have weakened the role of the server, or even abolish the server, allowing any nodes in the

network serve the role of server and client, and exchange data among these node, not through the intermediate server, so make full use of network bandwidth resources, and improve the network scalability.

3.1 Centralized directory model of P2P networks

For example, Napster[25] (Figure 1), its network model is similar to the traditional Client/Server network model, the central index server in this model will be mainly responsible for the user request and return the node address with the aimed resource to the users. The detail data exchange process are directly communicate by these two nodes, and not through other server. The network model has high efficiency in resource search, but when the size of the network expand to a certain extent, the central index server will often become the performance bottleneck of the networks, and the entire network will slide into paralysis when the index server fail.

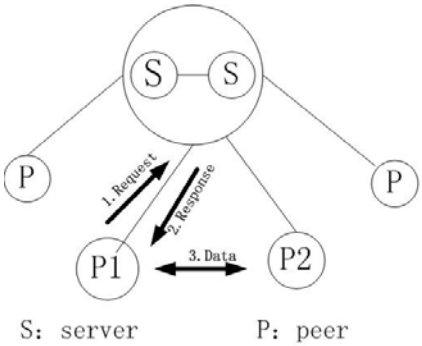

Figure 1 Centralized directory model

3.2 Pure distributed model of P2P networks

For example, Gnutella [26] (Figure 2), it is a pure distributed p2p network model, which use the flooding mechanism to search the resource of the network. The node contains the aimed resource that users need will send response message to the source request node, and then these two nodes directly exchange this resource, this model do not lead to the paralysis of the entire network because of the failure of someone node in the network, but the flooding search mechanism have poor efficiency, and will bring a lot of power exponential growth of the request message number during the process of the resource search, which cause serious load to the network. From figure 2, it can be seen that the middle node have heavy load, if not to reduce the load of the middle node, it's very possible that the time request this node to deal with will become a long delay.

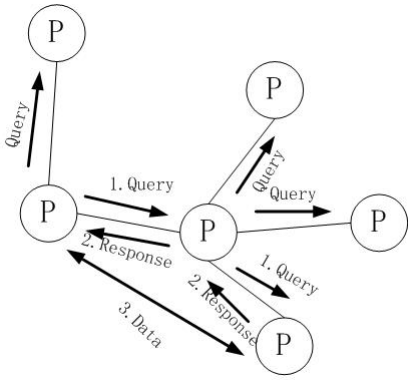

Figure 2 Pure distributed model

3.3 Based on super-node model of P2P networks

This model not only takes into account the high search efficiency of centralized directory model, but also considers the robustness of the pure distributed model. This model as shown in Figure 3. Although the request of the nodes in the cluster can process through the super-node in the cluster, decrease the cost of the network resources caused by the flooding message request, but there will make the super-node to become the bottleneck of the network if there will be a lot of request in the cluster.

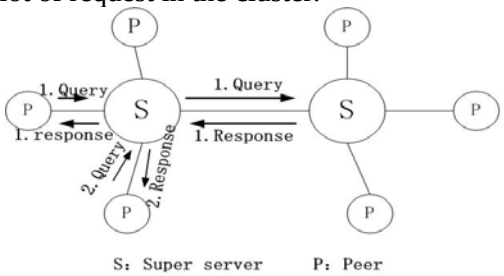

Figure 3 Based on super-node model

4 Replication Mechanism

From the above three P2P network model, we will see that there may be caused the loss or not available of some important resource in the network because some important nodes have heavy load or suddenly fail. So we should introduce replicate mechanism to the P2P network, backup some important data in some other nodes in the network to decrease the loss of the important data and the load imbalance of the node in the network, transfer some load from the nodes with heavy load to other nodes with light load, make the entire network's load to balance.

4.1 Classification of the replication mechanisms

Replication mechanisms can be classified by the reason of the causing, divide into the replication mechanism triggered by the resource interactive

process and replication mechanism triggered by some heavy load nodes need to transfer some load to other nodes. The replication mechanism triggered by the resource interactive process[27] can be divided into three categories:

- Replication mechanism based on the source request node: only store the resource that request node need on the source request node, this method is relatively simple to achieve. It will not bring a great deal of network loads, but it is not obvious for the improvement of the network performance.
- Unify replication mechanism based on the path: store the resource on all these nodes of the path, which is from the source request node to the target node providing the needed resource. The effect of this mechanism is clear, which can significantly reduce the delay time that request node get the needed resource, and at the same time greatly improve the fault-tolerant of the network, but this mechanism bring a great deal of network load to the entire network.
- Replication mechanism between the above two extreme ways, that is, in the path from the request node to the target node, this mechanism will use some algorithm to determine part of the all node along the path to store the needed resource. This mechanism is very flexible, not only obtain relatively good average delay time of user getting the needed resource, but also not bring comparatively heavy network load to the entire network.

4.2 Implementation of the replication mechanism

Replication mechanism based on the source request node and unify replication mechanism based on the path are extreme mechanism, so we use third replication mechanism in the general application, namely select certain algorithm to determine some nodes along the path to store the resources's backup. During the implementation process of replication mechanism, how to select the nodes that backup the needed resources is the key problem, as well as the number of these nodes. If a node will become the node backup resources or not, it has related with the current load, on-line time, capacity of the cache, the ability to deal with. How to determine the replication node is NP problem, so the algorithm determines what factors should be considered based on the specific need in the actual application process.

4.2.1 Node load

Node load is directly with the processing ability of this node, so when testing the actual load of the node, we send a request message with TTL=1 to this testing node through the part neighbor of the testing node, after the testing node receive this request message, the node will process and response this request message, then the neighbor nodes received the response message from this testing node, calculate the time difference between the time of sending the request message and the time of receiving the response message to construct the collection of the time difference, and select the minimum value of the time difference act as the load of the testing node, during the actual process, we have better select some time durations to test the minimum of these time difference. Although this time, including the processing time of CPU and the delay of network transmission, but to a certain extent the time difference can be a rough measure of the current node load. After knowing the all nodes load in the entire network, we should calculate the current average load value--Avg_{loan}, at the same time every node have to set up a data table, which record the current network load of the all neighbors nodes.

4.2.2 Selection of the replication node

Selection of the replication node is the core of the whole replication mechanism, because it needs to consider a lot of factors. There are many heuristic algorithms[28]: such as, the random heuristic algorithm, which use a random generator to determine the node of store resources according to a uniform distribution probability, this algorithm does not consider the current load of the node, as well as the on-line time of the node. HighlyUpFirst[29] heuristic algorithm, in accordance with the length of the on-line time of the node to determine the replication node, But the HighlyAvailableFirst[29] heuristic algorithm use the available information of the node to determine the replication nodes. Although these heuristic algorithms are relatively simple, these algorithms can improve the performance of the network to a certain extent.

On the assumption that we have already mastered all the network load of the nodes in the network, here we select replication nodes based on the network load of the node first, the cache size of the node second, and the restriction total number of replication to determine what nodes should be replication node.

4.2.3 Implementation of the replication algorithm

When these nodes having the needed resources receive the request message from the request node, go to process, then return corresponding response message, this message carry in backup data and return along to the original path. When this message arrival at one node, replication algorithm determine

whether this node is fit for the replication node or not according to the network load of this node, if the load value of this node is less than the given maximum load value $MAX_{loan,}$ then this algorithm consider the left size of storage cache of the node. if the left storage cache of the node is bigger than the size of the backup data, this algorithm will put the backup data carried by the message onto the node, then go to return this message, until reach the given total number of the replication—MAX_{copy}, or this response message arrival at the source request node to complete the implementation of the replication algorithm.

During the implementation process, when one node is determined to act as the replication node, and save the backup data, this replication mechanism need to record the source node address having the source backup data in the local data table, and return a message to the source node to ask that the source node update the backup statistics data table. When there are same request for certain resource sent to the source node next time, then the source node will match some data item with the data statistics table, and return all nodes address list, including the source data and backup data, the request node select the nearest data node to interact according the data table. This data table will be used to maintain the data consistency between the source data and backup data in the next section.

4.3 Load transfer[29]

The load value of every node will be changed at any time, when the load value of the node change, the node will send this newly load value to its neighbors to ask for updating the load value table of its neighbors. When some nodes's total load value is bigger than the given maximum load value, namely these nodes are overload. In order to make these overload nodes to normal handle with the request message, the replication algorithm should go to transfer the network load, which is from heavy load nodes to light load nodes, finally ensuring that the network have high implementation efficiency.

Load-transfer process: According to the neighbor nodes load data table stored in the node, the node select the minimum load node from all the nodes in the load data table, then send a inquiry message to this node with the minimum load value to ask for the backup resources list existing in this node. The minimum load node deal with the request, return all backup resources list of this node to the source request node, which receive the message, and compare the local resources list with the backup resources list of the message, then random select some resources that the minimum load node have not, according to the information of these resources to produce a load transfer request message, and send to the minimum load node. The minimum load node receive the load transfer request message, use the above node selection algorithm to judge whether the added load total is bigger than the given maximum load value and the local cache is bigger than these backup resources needed to transfer. If these conditions are satisfied, the minimum load node store the backup resource. At the same time this node get the source node information from the backup data through this message, then send the newly update message to the source node of the backup resources to ask for the source node to update its backup resources data table. If there are this backup resource message request sent to the source node in the next time, the source node will return a newly data address list, namely some heavy load nodes transfer some resource to others light load nodes, so for some resources request change from the heavy load nodes to light load nodes, thus decrease the load information of the source node, from Figure 4, at first there is only node F, which have certain resource, while in the network the other node want to get the resource, they must send a resource request message to node F for this resource through node G,E and I, node F have heavy load because of a lot of request, so we should select the current minimum load value node I to act as the node implementation of load transfer from all its neighbor nodes, then implement the transfer of load through the replication mechanism. When node I have this backup resource, these nodes that firstly ask node F for this resource or near node I can redirect to node I to get the needed resources, so part of the load of node F transfer to the node I. At the same time the load of node I change, so node I will send the newly local load value information to all the neighbor nodes to update their local load data table. If the load value of node F is bigger than the given maximum load value MAX_{loan}, node F will go to do the similar operation of transferring network load.

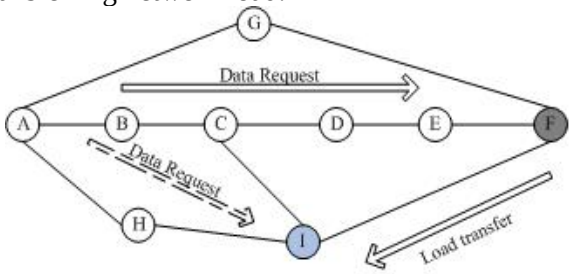

Figure 4 Load transfer

When the cache of the node is full, this node is impossible to store the new backup resource

according to the previous replication algorithm, but part of the backup resource stored on this node maybe have small visitors, so it is necessary to replace or clear these backup resources not often used to save a lot of cache to store those backup resources often used. Now there are some mature replacement algorithm, such as LRU (Least Recently Used), LFU (Least Frequently Used). When some backup resources are replaced in some nodes, these nodes will match these replacement backup resources with the backup resource data table, find the source node that have this backup resource, then send the request message to this source node, ask for the source node to clear the corresponding record of the replacement resource from the data table, and also remove the backup data record that have already removed from the cache in the local backup data table.

4.4 Performance analysis of replication mechanism

4.4.1 replication placement model

To investigate the sensitivity of replication placement performance to client locations, we look into several client replication placement models. Our goal is not to explore all possible client replication placements, but to consider the extreme cases, along with the random case, because the extreme cases can give us the boundary of expected performance.

The first model we look into is the random client replication placement, where the client nodes are selected at random with uniform probability.

We also look into the extreme client replication placement as defined in [30], namely extreme affinity and extreme disaffinity. The extreme affinity model places the clients replication as close as possible to each other; the extreme disaffinity model places the clients replication as far as possible from each other. The particular algorithm we use to place a number of clients replication on a graph according to the affinity/disaffinity model is described in [31]. Below is a brief summary of that algorithm. The first client replication is selected at random among all nodes. Then, we assign to each node n_i that is not selected yet the probability

$$p_i = \frac{\alpha}{w_i^{\beta}}$$

where w_i is the closest distance between node n_i and a node that is already selected as a client, α is calculated such that $\sum_{n_i} p_i = 1$, and β is the parameter that defines the degree of affinity or disaffinity. After a node is chosen to be a client, the probabilities of the remaining nodes are recomputed and the process is repeated until the desired number of clients is selected. Similar to [31], in our experiments we use β = 15 and β = -15 for extreme affinity and disaffinity respectively.

To verify our results with real-world data, we use web server access logs to create the population of clients. In particular, we collect the unique IP addresses of all clients that have accessed the same Web server within some period of time. Then, we run a traceroute to each of the client addresses. Finally, we intersect each of the traceroute paths with the Internet map to find the last-hop router toward a Web client that is on that map. The set of all last-hop routers is our web clients set that can be used to represent the population of the real-world Web clients.

4.4.2 Performance analysis

Here we use the special two parameters to measure the performance of the network, such as the average customer latency and the overall network overhead. For the convenience to analysis, assuming that the latency between the two nodes has related with the hop between the two nodes (in the actual network, latency is directly related with the hops of the route layer), at the same time we assume that all the network bandwidth among these nodes are same (in the real networks, all nodes have the same network bandwidth is impossible). In these two premise, we can get the average client latency value AveClientLatency,

$$AveClientLatency = \frac{\sum_{client(c)} Dist(c, replica(c))}{NumberOfClients} \quad (1)$$

where replica(c) is the replication node for client c, and Dist(c, replica(c)) is the distance between them in number of hops.

The total network load have divided into two part, such as the load caused by transferring the source resource from the source node to all the replication node, and the load caused by transferring the backup data from the backup node to all the user node. But in fact, the number of backup nodes for saving these source resources are limited, so the load caused by transferring the source resource from the source node to all the replication node is far less than the load caused by transferring the backup data from the backup node to all the user node. Here we rough regard put the latter network load as the total load of the network, but the latter network load is directly related with the hops from every client to the

replication nodes of object resource, so the total network load can be seen as

$$NetworkOverhead = \sum_{clients(c)} Dist(c, replica(c)) \quad (2)$$

In our analysis, we are not interested in the absolute client latency or absolute network overhead parameter. Instead, we are interested in the relative client latency or relative network overhead. Based on our assumptions, we have

$$AveClientLatency = \frac{NetworkOverhead}{NumberOfClients} \quad (3)$$

In Figure 4, when node F transfer part of load to node I, we can calculate that the average network latency after the load transfer is significantly less than the average network latency before the load transfer. The larger the network, the improvement of network performance will be better through this replication mechanism.

5 Consistency solution of replication mechanism

At present, many P2P networks access the network primary based on the read operation, so it will not be involved in data synchronization or data consistency, but in the future P2P applications will broad across every walk of life, including e-commerce, coordination process. These applications not only need the read operation, but also need the write operation to complete the data coordination and synchronization, so how to maintain the data synchronization or consistency between the source data and backup data is the key to the expansion of the P2P network in the future.

At present, there are mainly three solutions to maintain the network data consistency:

① Active way, using the push method. When the data of the source node have changed, the source node will put this update message to broadcast in the network, after all the nodes of the networks receive the message, these nodes determine if there exist data backup resource through the operation of match, if exist the resource, go to download and update. This update operation is launched from the source node, more suitable for application in the static networks. The advantages of this way is that all the backup data will be timely update, and decrease the latency time of user request for the newly backup data, but the defect of this way is the broadcasting data update, this mechanism will make the load of network become heavy, and seriously effect the communication quality of the network.

② Passive way, using the poll method, after the source node modify the data, it do not directly send the update message to others nodes to tell them to update the data resource of the source node, but only modify the version of the data related with the source data node. When others nodes having this backup data need to handle with these data, the program will take out the source node having this backup data from the local data table, then send a request message, which carry on the version of corresponding data in the current node. When the source node receive this message, find that the data version in replication nodes is not the newest, the source node will send the newest data resource to this backup node according to the data modification information among the version of the data to complete the update operation. This way is launched from the backup node, comparatively suit for the dynamic network environment. The advantages of this way is that do not immediately update, only go to update when the user need to use this data, but the defect is that the latency that user get the newest update data will become longer.

③ Push & poll way, a mixed approach, considering the advantages of the push way and the poll way, when the source data node update the data in the node, the source node will send update message to part of other nodes having this backup data, and ask for these nodes to immediately update. When there are some part of these nodes have not update these data, so these nodes will automatic link to the source data node during a period of time, determine whether the data should update or not through compared with the version number, and further to deal with. Of course, the time intervals will change along with the change of the network data, if the network data change quickly, the time interval will be become shorter.

```
While (the update queue of the data source node have update message, namely, the request queue is not empty)
/ * Through the queue to maintain the revised data synchronization * /
{
Get the head element of the update queue;
If (the head element is the source node (on behalf of the party to launch update operation))
{
  Update the source data in the source node, and update the version of the source data;
  Random select a subset of the update queue; / / the improved push algorithm
  Send update message to all the nodes in the subset;
  Extract the version carried in the response message returned from the subset;
  Determine which part of data has been changed through the comparison between these different data version;
  Send the changed data blocks to the backup node;
}
Else (the head element is the backup node)
{
   The backup node send the backup data version to the source data;
   The backup node determine whether to update the data of the current node according to the message returned from the source node;
   / * The detailed steps of this operation: the source node receive the data version of the backup node, and match the version number. If the version number is not the same, the source node determine what data block have already modified in this version, then send the modified data block to backup node to update; If the version number is same, it is shown that the data in the backup node is the newest data, so it is not necessary to update the data.* /
   The backup node update the data source; / * update operation launched by the backup node. * /
   The backup node send the update data to the source node to update the data in the source node, and update the data version, and then recursive select a subset to update data based on the source node.
 }
```

Figure 5 Data consistency algorithm

Reference[32] provide that data consistency is built on the above premise: the resources can only be modified through the owner of the document to implement. But this premise is not feasible in the co-processing applications of the actual network, so here we provide a newly improvement data consistency, data can be modified in any backup node, but the modified data of the backup node should synchronize with the data of the source data node. Here we use the above 3 way to maintain data consistency, the algorithm is described as shown in Figure 5, the algorithm use the improved push and pull methods. Among which the improved push method require the source data node to select a subset of the backup data set from the data table, then send the update message request to all the nodes of the subset. The improved pull method ask the backup node to automatic link to the source data node to update the data resource during a period of time. Through this algorithm, it does not increase the network load due to many data modified operation of the backup node, and it can decrease the latency of getting data. This algorithm introduce the update queue of the data source node to ensure the data synchronization during the data modify process.

6 Conclusion

This paper introduce replication mechanism to the P2P networks, not only solve the loss of the important data because of the failure of some important node in the network, and through the load transfer operation among the nodes this replication mechanism make the P2P networks load to balance, efficient use the resources of the P2P network, and ensure the update data synchronization of the P2P network through the data consistency algorithm, further promote the commercial application of the P2P networks.

References:

[1] Jaime Lloret, Juan R.Diaz, Jose M.Jimenez, Manuel Esteve. The Popularity Parameter in Unstructured P2P File Sharing Networks. WSEAS TRANSACTIONS on COMPUTERS, vol. 3, pp. 2118-2133, December 2004

[2] M. Sasabe, N. Wakamiya, and M. Murata. Adaptive and Robust P2P Media Streaming. WSEAS TRANSACTIONS on COMMUNICATIONS, vol. 4, pp. 425 − 430, July 2005.

[3] Zheng Yan and Peng Zhang. Trust Collaboration in P2P Systems Based on Trusted Computing Platforms. WSEAS Transactions on Information Science and Applications, Issue 2, Vol. 3, pp. 275-282, February 2006.

[4] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In Proc. of the 16th annual ACM International Conf. on Supercomputing (ICS ' 02), New York, USA, June 2002.

[5] J. Kangasharju, K.W. Ross, and D. Turner. Optimal Content Replication in P2P Communities. Manuscript. 2002.

[6] E. Cohen and S. Shenker. Replication Strategies in unstructured peer-to-peer networks. In Proc. of ACM SIGCOMM ' 02, Pittsburgh, USA, Aug. 2002.

[7] Haifeng Yu and Amin Vahdat. Minimal Replication Cost for Availability. In Proc. of the 21th ACM Symposium on Principles of Distributed Computing (PODC), July 2002.

[8] Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In Proc. of ACM SIGCOMM ' 01, San Diego, USA, Aug. 2001.

[9] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Proc. of the IFIP/ACM International Conf. on Distributed Systems Platforms, Oct. 2001.

[10] J. Lan, X. Liu, P. Shenoy, and K. Ramamritham. Consistency maintenance in peer-to-peer file sharing networks. Proc. IEEE Workshop on Internet Applications, pp.90-94, 2003.

[11] A. Datta, M. Hauswirth, and K. Aberer. Updates in highly unreliable, replicated peer-to-peer systems. Proc. ICDCS ' 03, pp. 76-85, 2003.

[12] Y. Huang, P. Sistla, and O. Wolfson. Data replication for mobile computer. Proc. ACMSIGMOD ' 94, pp.13-24, 1994.

[13] S.K. Madria. Timestamps to detect R-W conflicts in mobile computing. Proc. Int ' l. Workshop on Mobile Data Access (MDA ' 98), pp.242-253, Nov. 1998.

[14] E. Pitoura and B. Bhargava. Maintaining consistency of data in mobile distributed environments. Proc. IEEE ICDCS'95, pp.404-413, 1995.

[15] E. Pitoura, P.K. Chrysanthis, and K. Ramamritham. Characterizing the temporal and semantic coherency of broadcastbased data dissemination. Proc. Int'l Conf. on Database Theory (ICDT ' 03), pp.410-424, 2003

[16] G. Cao, L. Yin, C.R. Das. Cooperative cache-based data access in ad hoc networks. IEEE Computer, Vol.37, No.2, pp.32-39, 2004.

[17] L.D. Fife and L. Gruenwald. Research issues for data communication in mobile ad-hoc network database systems. SIGMOD Record, Vol.32, No.2, pp.42-47, 2003.

[18] K. Wang and B. Li. Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks. Proc. IEEE Infocom'02, Vol.2, pp.1089-1098, 2002.

[19] G. Karumanchi, S. Muralidharan, and R. Prakash. Information dissemination in partitionable mobile ad hoc networks. Proc. SRDS ' 99, pp.4-13, 1999.

[20] J. Luo, J.P. Hubaux, and P. Eugster. PAN: Providing reliable storage in mobile ad hoc networks with probabilistic quorum systems. Proc. ACM MobiHoc 2003, pp.1-12, 2003.

[21] K. Rothermel, C. Becker, and J. Hahner. Consistent update diffusion in mobile ad hoc networks. Technical Report 2002/04, Computer Science Department, University of Stuttgart, 2002.

[22] D. Barbara and H. Garcia-Molina. The reliability of vote mechanisms. IEEE Trans. on Computers, Vol.36, No.10, pp.1197-1208, 1987.

[23] D. Malkhi, M.K. Reiter, and A. Wool. Probabilistic quorum systems. Information and Computation, Vol.170, No.2, pp.184-206, 2001.

[24] Z.J. Haas, J.Y. Halpern, and L. Li.Gossip-based ad hoc routing. Proc. IEEE Infocom 2002, pp.1707-1716, 2002.

[25] Napster Website[EB/ OL] . http://www. napster. com

[26] Gnutella Website[EB/ OL] . http://www. gnutella. com

[27] Yamamoto, H.; Maruta, D.; Oie, Y..Replication methods for load balancing on distributed storages in P2P networks.Applications and the Internet, 2005. Proceedings. The 2005 Symposium on 31 Jan.-4 Feb. 2005 Page(s):264 - 271

[28] Drougas, Y.; Kalogeraki, V.A fair resource allocation algorithm for peer-to-peer overlays.INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE Volume 4, 13-17 March 2005 Page(s):2853 - 2858 vol. 4

[29] On, G.; Schmitt, J.; Steinmetz, R..The effectiveness of realistic replication strategies on quality of availability for peer-to-peer systems.Peer-to-Peer Computing, 2003. (P2P 2003). Proceedings. Third International Conference on1-3 Sept. 2003 Page(s):57 - 64

[30] Graham Phillips, Scott Shenker, and Hongsuda Tangmunarunkit. Scaling of Multicast Trees: Comments on the Chuang-Sirbu scaling law. In Proceedings of the ACM SIGCOMM'99, Cambridge, Massachusetts, USA, August 1999.

[31] Tina Wong and Randy Katz. An Analysis of Multicast Forwarding State Scalability. In Proceedings of the 8th IEEE International Conference on Network Protocols (ICNP 2000), Osaka, Japan, November 2000.

[32] Jiang Lan; Xiaotao Liu; Prashant Shenoy; Krithi Ramamritham. Consistency maintenance in peer-to-peer file sharing networks.Internet Applications. WIAPP 2003. Proceedings. The Third IEEE Workshop on 23-24 June 2003 Page(s):90 - 94