Reducing Data Replication Overhead in DHT Based Peer-to-Peer System

Kyungbaek Kim and Daeyeon Park

Department of Electrical Engineering & Computer Science,
Division of Electrical Engineering,
Korea Advanced Institute of Science and Technology (KAIST),
373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea
kbkim@sslab.kaist.ac.kr, daeyeon@ee.kaist.ac.kr

Abstract. DHT based p2p systems are appeared to provide scalable storage services with idle resource from many unreliable clients. If a DHT is used in storage intensive applications where data loss must be minimized, quick replication is especially important to replace lost redundancy on other nodes in reaction to failures. To achieve this easily, the simple replication method directly uses the consistent set such as the leaf set and the successor list. However, this set is tightly coupled to the current state of nodes and the traffic needed to support this replication can be high and bursty under churn.

This paper explores efficient replication methods that only glimpse the consistent set to select a new replica. We propose two types of replication methods : *Quorum based replication* and *Availability based replication*. The replicas are loosely coupled to the consistent set and can eliminate the compulsory replication under churn. Results from a simulation study suggest that our methods can reduce network traffic enormously and achieve high data availability in a DHT based p2p storage system.

Keywords: Peer-to-Peer, Replication, Data availability.

1 Introduction

In these days, peer-to-peer systems have become an extremely popular platform for large-scale content sharing, even the p2p based file systems appear. Unlike client/server model based storage systems, which centralize the management of data in a few highly reliable servers, peer-to-peer storage systems distribute the burden of data storage and communications among tens of thousands of clients. The wide-spread attraction of this model arises from the promise that idle resources may be efficiently harvested to provide scalable storage services. To promise that, a lot of research papers discussed the Distributed Hash Table (DHT) based p2p routing algorithms [1] [2] [3] [4] and we call the p2p system which uses the DHT based p2p routing algorithm the structured p2p system.

These structured p2p systems achieve the efficient and bounded lookup for the requested object. However, they poorly support the acceptable levels of data availability [11] [12]. The main problem is the ad hoc manner in which p2p systems are constructed. In contrast to traditional systems, peer-to-peer systems are composed of components with extremely heterogeneous availabilities - individually administered host

PC's may be turned on and off, join and leave the system, have intermittent connectivity, and are constructed from low-cost low reliability components. For example, one recent study of a popular peer-to-peer file sharing system[7] found that the majority of peers had application-level availability rates of under 20% and only 20% nodes have server-like profiles. In such an environment, failure is no longer an exceptional event, but is a pervasive condition. At any point in time the majority of hosts in the system are unavailable and those hosts that are available may soon stop servicing requests.

Many structured p2p systems use the simple replication method to cope with the massively failures of nodes [5] [6] [8]. This simple approach exploits the consistent set of the successive nodeIDs such as successor list of chord [1] and leaf set of pastry [2]. Basically, these sets are used to conserve the routing information to cope with the failures and when any node joins or leaves, the consistent set of every node which detects the change of the membership must update to preserve the current state of the p2p system. Because of this consistent and automated update, many systems use this set to replicate the responsible objects of each node for the simple lookup and the simple data management. However, this simple approach makes high network traffic because of the dynamic membership of the p2p system. For example, if the size of the consistent set is 8 and the p2p system needs 6 replicas for the target data availability, when one new node changes its state, the 8 nodes which are the members of the consistent set of the new node should update their consistent sets. In this case, the simple replication approach is tightly coupled to the consistent set and 6 replicas should be updated without any relation of the current data availability or the node characteristics. According to this behavior, the heavily dynamic membership change of p2p systems causes the compulsory data replication and generates very high network traffic for this replication. Because of this heavy replication overhead, until now, DHT algorithms are not widely used in commercial systems yet.

In our paper, we suggest the efficient replication methods to achieve the highly durable p2p storage system with small maintenance cost. The replicas are loosely coupled to the consistent set and they are interleaved on the consistent set to reduce the compulsory copies which occur under churn. The method with this concept is called the Quorum based replication. In this replication, each node keeps the number of replicas more than the target quorum to achieve target data availability. Moreover, we exploit the node availability and select more reliable nodes as replicas to delay the replication and to reduce the network cost. This Availability based replication calculates the data availability whenever the consistent set changes and guarantees the high data availability by the numerical value. This replication should predict the node availability of each node. To do this, each node manages its availability and advertises it to all members of the consistent set by piggybacking it to the periodic ping message which has been already used to detect node failures on the consistent set.

Our replication methods need additional information for replicas and interleave the replicas. Unlike the simple replication, sometimes, each node does not have the objects which are serviced by the right next neighbors such as successor or predecessor of chord [1]. In this case, each node should contain the information for the replicas of all members on the consistent set to guarantee the correct data routing whenever any node fails or leaves. Because of this complication of replication methods, subtle data management should be needed under churn.

We evaluate the effect of our replication method for the p2p system by using an event driven simulation. We compare the network traffic for various target availability and various node characteristics between the simple replication and the proposed replication. We show that our methods enormously reduce the network traffic to achieve the same target availability.

The rest of this paper is organized as follow. Section 2 briefly presents the DHT based p2p system and problems of the simple replication. Section 3 presents our proposed replication methods for the durable p2p storage system. The performance evaluation is on section 4. We mention other related works in section 5. Finally, we conclude this paper on section 6.

2 Background

There are many DHT based p2p algorithms such as chord, pastry, tapestry and can. Each node in the DHT based p2p system gets a 128-bit node identifier (nodeID). The nodeID is used to indicate a node's position in a circular ID space and it is assumed that nodeIDs are generated such that the resulting set of nodeIDs is uniformly distributed in the 128-bit ID space. Each node is responsible for storing and servicing the objects which are on the range between its node and a neighbor node. This object range of a node changes dynamically under churn. Assuming a network consisting of N nodes, the DHT based p2p system can route to the numerically closest node to a given object key in less than $O(log_2N)$ steps.

These algorithms can lookup any data efficiently with DHT, but when the massively node failures occurs and spoils the information of DHT without any notification, this efficient lookup can not guarantee the correctness. To cope with the massively node failures, they use the consistent set such as the successor list of chord and the leaf set of pastry. This consistent set of a node is composed of the neighbor nodes which locate numerically near to the node on ID space. This set is tightly coupled to the current state of nodes and when any node joins or leaves, the consistent set of every node which detects the change of the membership must update to preserve the current state of the p2p system. The p2p system guarantees the correctness unless all members of consistent set fail simultaneously.

This consistent set is used for not only the routing correctness but also the data availability on durable p2p storage systems such as p2p file systems [5] [6] and p2p file sharing systems [8]. Data Availability means the total availability when the multiple nodes have the data. This availability is obtained by subtracting the probability of that all nodes which have the data leave from 1. That means if only one node is alive, the data is available. Like the figure 1(a), one node replicates stored objects to the neighbor nodes which are the member of consistent set until the replicas are enough to achieve target data availability. This simple replication guarantees the simple data availability management and the simple lookup under churn easily and automatically. Because the consistent set has the current state of nodes and updates immediately under churn, the p2p system keeps the target data availability automatically. Moreover, because neighbor nodes of a node already have the replicas of its objects, even if this node leaves, the

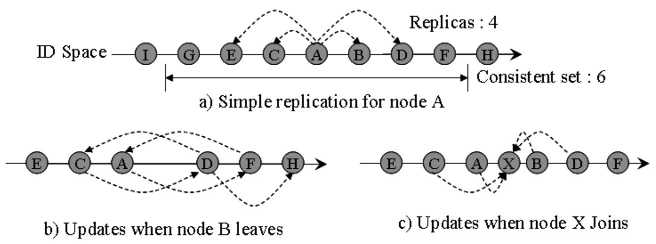

Replicas : 4
ID Space
I
G
E
C
A
B
D
F
H
Consistent set : 6
a) Simple replication for node A
X
b) Updates when node B leaves
c) Updates when node X Joins

Fig. 1. Simple replication in DHT based p2p

neighbor node automatically replaces it as a servicing node for its object range without additional object copies.

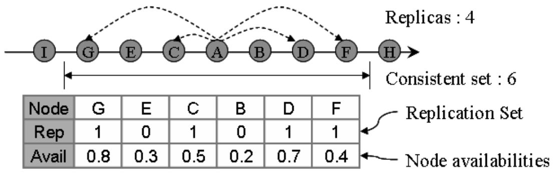
However, the simple replication causes the more maintenance traffic under churn. If the number of replicas is N and a node leaves, the new $N + 1$ replicas are needed for the affected nodes. In figure 1(b), when a node B leaves, the nodes A, C, D, F which already have the replicas on node B should make new replicas and node D which is newly responsible for the object range of node B makes the replica for this range additionally. Moreover, when a node joins, each affected node copies the objects to it as a new replica like figure 1(c). In this case, when node B joins and leaves very frequently, the compulsory data replications occurs and the heavy data traffic wastes even if the dynamic behavior of node B can not affect the data availability. According to this simple behavior and the heavy churn of the p2p participants, the data traffic needed to support the simple replication is very high and bursty.

3 Proposed Idea

3.1 Quorum Based Replication

The simple replication method basically uses the concept of the quorum. The quorum means that the fixed minimum number of members of a set which must be present for its objective to be valid. That is, if the number of the replica for an object is more than the target quorum, the p2p system considers that the object is available under massive failures. However, this simple method directly uses the consistent set such as a successor list or a leaf set which is tightly coupled to the state of the current network. Under churn, to keep the right information of the network, the affected node should update its consistent set and the members of this set change very dynamically. Consequently, the simple replication method is affected by the change of the consistent set and needs too much traffic to keep the availability of an object. Sometimes, this compulsory copy for the replica is meaningless to the availability because the new replica leaves soon.

To prevent this compulsory copy, we modify the replication method which is loosely coupled to the consistent set. Like the figure 2, we add the new information; the replication set that indicates which node replicates the object. The range of this set is same to the consistent set, but the update of this set occurs individually. Like the simple method, the replication only occurs when the number of replicas is fewer than the target quorum. However, if the leaving node is not a member of the replication set, there is no need to



Node	G	E	C	B	D	F
Rep	1	0	1	0	1	1
Avail	0.8	0.3	0.5	0.2	0.7	0.4

Fig. 2. Metadata for our replication methods

find a new replica and the p2p system can reduce the compulsory copies. When a node needs a new replica, it selects the numerically closest node from it, because the edge of the consistent set may change more easily and more frequently than the inner side.

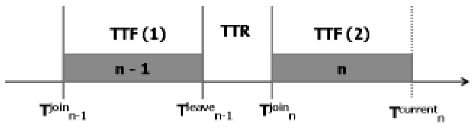
When a new node joins, it gets not only routing information such as DHT and consistent set but also the replication set. Unlike the simple method, a new node already knows the information of replicas and the data copy only occurs for the object range which is responsible for it. Other nodes whose consistent sets are affected by the new node check whether the replication is needed and if it is, the node makes a new replica. However, because in the general DHT p2p the size of the consistent set is bigger than the number of replicas, the replication does not occur frequently and the replication set can interleave the replicas on the consistent set. This behavior increases the chance to reduce the compulsory copies.

3.2 Availability Based Replication

The quorum based replication considers that each node has the same availability and it tries to keep the number of replicas above the target quorum to achieve the target data availability. However, if a new replica is assigned by the node which has low availability, this node may leave soon and we need another new replica. If we select a new replica with the node availability, we can select the more available node as a replica and can reduce the overhead. To achieve this, the consistent set has the availability information of all members like figure 2.

We assume that the node availability is the prediction value how long a node is alive after it joins the p2p system, because in other research[7] the long lived nodes generally have the large bandwidth and the big computing power. The figure 3 shows this availability prediction mechanism. We use the Mean Time To Failure and the Mean Time To Recover to estimate the node availability. MTTF is the average value how long a node is alive after it joins and MTTR is the average value how long a node is sleep after it leaves. We can get MTTF and MTTR by using the last join time, the last leave time. Unlike MTTR, we periodically update the MTTF by using the current time and the join time because MTTF can change during a node join the p2p system. The average value of MTTF and MTTR is obtained by the sum of the weighted value estimation process. According to these values, we compute the node availability with the equation, MTTF/(MTTF + MTTR).

The availability information is computed by each node and each node advertises this information to all members of the consistent set by using the piggyback method. To detect the node failure, a node sends a ping message to all member of the consistent set.



Mean Time To Failure (MTTF)
- Time to failure (TTF)
  1) At joining measurement : $\mathbf{TTF_n} = T^{leave}_{n-1} - T^{join}_{n-1}$
  2) At periodic measurement : $\mathbf{TTF_n} = T^{current}_{n} - T^{join}_{n}$
- $MTTF_n = \alpha * TTF_n + (1-\alpha) * MTTF_{n-1}, (0 < \alpha < 1)$

Mean Time To Recover (MTTR)
- Time to Recover (TTR)
  – $\mathbf{TTR_n} = T^{join}_{n} - T^{leave}_{n-1}$
- $MTTR_n = \beta * TTR_n + (1-\beta) * MTTR_{n-1}, (0 < \beta < 1)$

Node Availability = MTTF / (MTTF + MTTR)

Fig. 3. Node availability prediction

We piggyback the availability information to this ping message and each node manages the consistent set with the node availability.

Like the quorum based replication, the availability based replication use the replication set to make that the replicas are loosely coupled to the consistent set. The main difference of these replications is the selection of a new replica. In this approach, the replication only occurs when the data availability is below the target availability. If the leaving node is not a member of the replication set, there is no need to replicate the data. Otherwise, if it is a member, we select the most available node among non-members of the replication set as a new replica.

When a new node joins, the basic operation is similar to the quorum based replication. The routing table, consistent set and replication set are copied and the other nodes whose consistent sets are affected by the new node decide whether they make new replicas. However, because the availability based replication takes care of selecting new replicas by computing the availability, if all members of a consistent set have averagely low availability, it need more replicas than the quorum based replication. Sometimes this behavior takes more bandwidth, but when nodes leave, this subtle replication can reduce much more bandwidth than the quorum based replication.

3.3 Management of the Replication Set

Unlike the simple replication, our replications interleave the replicas on the consistent set. When a node fails and its neighbor gets the lookup request, this neighbor may not have the replicas for the requested object. In this case, the neighbors must forward the request to the replicas of the failed node for the routing correctness. To do this, each node should have the replication sets of all members of its consistent set by piggybacking this information to the periodic ping message for its consistent set.

Moreover, we should consider that the change of the object range affects the replication set. When a new node joins and a target node gets this join request, the object range of the target node is divided into two object range and the new node is responsible for one of them. In this case, the new node simply copies the replication set and adds the target node as a new replica because it already has the object for

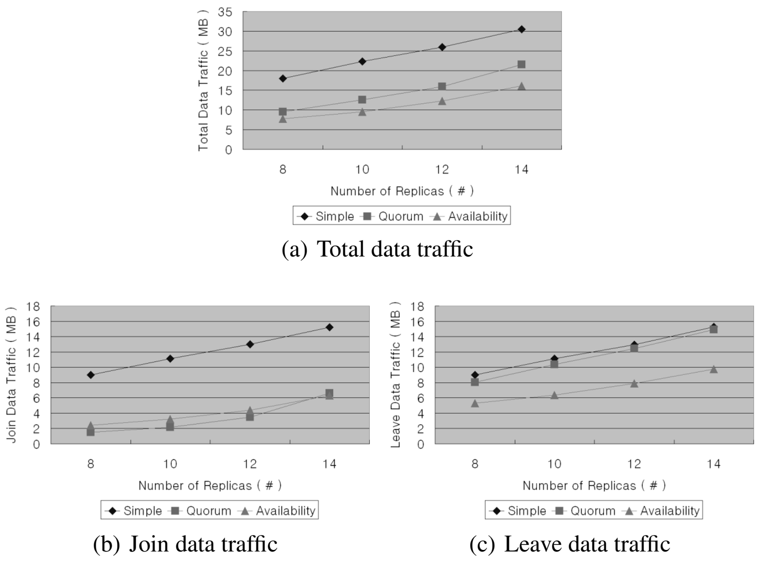


Fig. 4. Data Traffic with various number of replicas

this range. When a node leaves or fails, its neighbor node is responsible for its object range. In this case, both replication sets of the failed node and its neighbor node are merged.

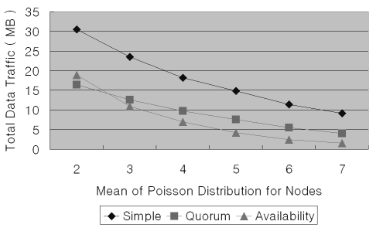
4 Evaluation

4.1 Simulation Setup

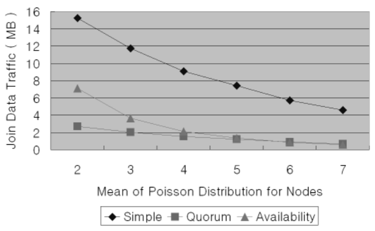
We make our p2p simulator which emulates the node behavior on the application layer. We implement the previous DHT based p2p algorithm, Pastry. We apply the simple replication and our new replications to the pastry. We use the 160 bit ID space and use 2000 nodes to organize the p2p system. The size of the consistent set is 16 and the number of replicas is variable from 8 to 14. The target availability for the availability based replication is decided by the number of replicas. We use the Poisson distribution to make the dynamic characteristics of nodes and use the exponential distribution to assign join/leave duration of a node. According to this Poisson distribution, 80% of total nodes have short lifetime and frequently join/leave and only 20% of total nodes have the reliable server-like profile. Recent research [7] measures the life distribution of the p2p nodes and its result is similar to our distribution, and we can tell that our distribution is similar to the real world.

4.2 Reduction of Data Traffic

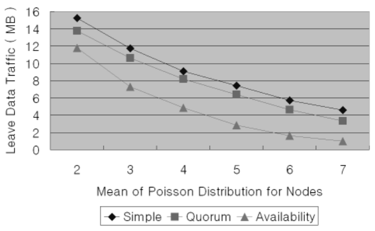
The figure 4 shows the comparison of average data traffic per a node with various replication methods. As we expect, the simple replication needs much more data traffic to



(a) Total data traffic



(b) Join data traffic



(c) Leave data traffic

Fig. 5. Data Traffic with various node characteristics

achieve the same data availability than our replications. The quorum based replication reduces the data traffic by about 40% and the availability based replication reduces the data traffic by about 60%.

To find out the detailed effect of our replications, we separate the join data traffic from the leave data traffic. When a node joins, the affected nodes update their replicas and we call the needed traffic the join data traffic. When a node leaves, the needed data traffic is called the leave data traffic. In the figure 4(b) and figure 4(c), the simple replication uses the similar amount of traffic for join and leave. The main reason is that the replication is tightly coupled to the consistent set and in both type of the changes, the similar amount of compulsory copies is needed to support the simple replication. However, in our two replications, the join data traffic is less than leave data traffic. The replication set is loosely coupled to the consistent set and when a join occurs, a node can decide which it makes a new replica. According to this behavior, the replicas are interleaved on the replication set and we can reduce the number of compulsory copies when a new node joins.

However, like figure 4(c), the quorum based replication needs similar amount of leave data traffic to the simple replication. The quorum based replication does not consider the node characteristics and some new replicas leave the system early after they are chosen by the other nodes. The figure 4(c) shows that the availability based replication solves this problem and needs less leave data traffic than the quorum based replication. On the other hand, the availability based replication computes the data availability whenever the membership changes. According to this, it takes more care of selecting new replicas and it needs more join data traffic than the quorum based replication until the number of replicas is similar to the size of the consistent set.

4.3 Effect of Node Dynamicity

The previous results show that our replications reduce the data traffic needed to achieve the high data availability. In this result, we try to find out the effect of our replications on the various node characteristics. The figure 5 shows the needed data traffic for each replication method when the mean of the Poisson distribution changes from 2 to 7. When the mean value increases, the average life time of a node increases. In this simulation, the target number of replica is 8. In the figure 5(a), when the mean value increases every replication method takes less data traffic, because there are more reliable nodes and they do not join/leave frequently.

As described on previous results, our replications can save more data traffic for any cases. We pay attention to the difference between the quorum based replication and the availability based replication. Generally, the availability based replication reduces more traffic, however when the mean value is 2, the quorum based replication saves more data traffic. The main reason of this fact is the join data traffic in figure 5(b). As we mentioned, the availability based replication takes more care of selecting the new replicas when a new node joins. When the most of nodes join/leave frequently this subtle care needs more replicas than the quorum and takes much more traffic.

5 Related Work

The commercial p2p file sharing systems leave the data replication up to the popularity of the data. The popular data is replicated on many clients and the data availability of this data is very high. However, the unpopular data are stored on few clients and it is very hard to find this data because of very low data availability. To make the p2p storage system durable, the smart data replication methods is needed and the each inserted data is available for any time and has the similar data availability.

In the paper[9], they stores the replicas on the random nodes on the ID space and periodically checks their availability. This behavior reduces the compulsory copy because the replication has no relation to the consistent set. However, this approach takes too much control traffic to keep the node availability of all replicas for every object on the system. Moreover, they do not use the consistent set and the change of the data availability caused by the node failure is detected slowly. The paper [10] shows that the erasure coding approach reduces the traffic of the replication by using the computing power. This approach is orthogonal to our approaches and we can use this coding with our replication methods.

6 Conclusion

We explore the efficient replication methods to make the DHT based p2p storage system more durable. Our replication methods are loosely coupled to the consistent set such as a successor list of chord and a leaf set of pastry and interleave the replicas on it. Because the consistent set updates the current state of nodes automatically, we can update the data availability immediately under churn. Moreover, we use the node availability to select more reliable replicas and we can reduce more data traffic when a node leaves.

According to these behaviors, the DHT based p2p storage system with our replication methods achieves the high data availability with small data traffic. This can encourage that the DHT based p2p algorithms are applied to the durable storage system.

References

1. I.Stoica, R.Morris, D.Karger, M.F.Kaashoek, and H.Balakrishnan. *Chord: a scalable peer-to-peer lookup service for internet applications*, In Proceedings of ACM SIGCOMM 2001, August 2001.
2. A.Rowstron and P.Druschel. *Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems*, In Proceedings of Middleware, November 2001.
3. B.Y.Zhao, J.Kubiatowicz, and A.Joseph. *Tapestry: An infrastructure for fault-tolerant wide-area location and routing*, UCB Technical Report UCB/CSD-01-114, 2001.
4. S.Ratnasamy, P.Francis, M.Handley, R.Karp, and S.Shenker. *A scalable content-addressable network*, In Proceedings of ACM SIGCOMM 2001, 2001.
5. P. Druschel and A. rowstron. *PAST: A large-scale, persistent peer-to-peer storage utility*, In Proceedings of HotOS VIII, May 2001.
6. F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica. *Wide-area cooperative storage with CFS*, In Proceedings of SOSP 2001, Oct 2001.
7. S. Saroiu et al. *A measurement study of peer-to-peer file sharing systems*, In Proceedings of MMCN 2002, 2002.
8. K.Kim and D.Park. *Efficient and Scalable Client Clustering For Web Proxy Cache*, IEICE Transaction on Information and Systems, E86-D(9), September 2003.
9. R. Bhagwan, K. Tati, Y. Cheng, S. Savage and G. M. Voelker. *Total Recall: System Support for Automated Availability Management*, In Proceedings of NSDI 2004, 2004
10. R. Bhagwan, S. Savage, and G. M. Voelker. *Replication Strategies for Highly Available Peer-to-peer Storage Systems*, In Proceedings of FuDiCo, June 2002.
11. C. Blake and R. Rodrigues. *High Availability, Scalable Storage, Dynamic Peer Networks : Pick Two*, In Proceedings of HotOS-IX, May 2003.
12. R. Bhagwan, S. Savage, and G. M. Voelker. *Understanding Availability*, In Proceedings of IPTPS 03, 2003.